ESTIMATION AND INFERENCE OF RANDOM EFFECT MODELS WITH APPLICATIONS TO POPULATION GENETICS AND PROTEOMICS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Kirsten E. Eilertson

August 2011

© 2011 Kirsten E. Eilertson

ALL RIGHTS RESERVED

ESTIMATION AND INFERENCE OF RANDOM EFFECT MODELS WITH APPLICATIONS TO POPULATION GENETICS AND PROTEOMICS

Kirsten E. Eilertson, Ph.D.

Cornell University 2011

In this dissertation I present two methodologies for estimation and inference of random effect models with applications to population genetics and proteomics. The first methodology presented, SnIPRE, is designed for identifying genes under natural selection. SnIPRE is a "McDonald-Kreitman" type of analysis, in that it is based on MK table data and has an advantage over other types of statistics because it is robust to demography. Similar to the MKprf method, SnIPRE makes use of genome-wide information to increase power, but is non-parametric in the sense that it makes no assumptions (and does not require estimation) of parameters such as mutation rate and species divergence time in order to identify genes under selection. In simulations SnIPRE outperforms both the MK statistic and the two versions of MKprf considered.

With the right assumptions SnIPRE may be used to estimate population parameters, and in chapter 3 we discuss the robustness of the method to the assumption of independent sites. I also propose a procedure for more precise estimation of the confidence bounds of the selection effect, and then apply our method to *Drosophila* and human-chimp comparison data.

PROWLRE, an empirical Bayes method for analyzing shotgun-proteomics data, is introduced in the final chapter. While a fully Bayesian implementation of this model is straightforward, the empirical Bayes implementation is more challenging. I present an EM algorithm designed for fitting this latent variable

model and then compare the results to the Bayesian estimation on simulated and synthetic data.

BIOGRAPHICAL SKETCH

Kirsten Eilertson was born and raised in Austin, MN. She graduated from Saint Olaf College in 2006, with a degree in Mathematics and a concentration in Statistics. In the Fall of 2006 she started in the MS/PHD statistics program at Cornell University in Ithaca, NY and began working with Carlos Bustamante in 2007. After graduating in 2011 she will begin working for the Bioinformatics Core at the Gladstone Institutes in San Francisco, CA.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Carlos Bustamante, who has been an invaluable resource for ideas, motivation, and support. His counsel and encouragement, as well as the cooperation and camaraderie he encouraged in his lab have helped me to develop and expand my knowledge of statistics and to be conscientious in its application in biological research.

I would also like to thank Jim Booth. I have had the benefit of taking several classes from him as well as working with him on various parts of my research. His abilities as a teacher and collaborator are something I aspire to in my career as a statistician.

I am very lucky to be able to count many of my grad school classmates as close friends. A special thanks to Caitlin, Dave, Raj, Ben, Darcy, and Jeremiah. I have learned a lot from them while here, and could not have done this without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This dissertation proposes estimation and inference procedures for mixed effect models with specific application to analyses conducted in population genetics and proteomics. These methods are referred to as SnIRPE and PROWLRE, respectively. We show that these methods offer a substantial increase in power over traditional methods of analyses and model the data in a meaningful and highly interpretable manner. Random effects are often used to account for nuisance variables or over-dispersion, however in these analyses they play an important role in subject specific inference. While the applications discussed are distinct, there are a few unifying themes. These include the structure of the data and the goals of the analyses. In each application, Bayesian and empirical Bayes implementations are presented.

In both applications the goals of the analyses call for conclusions on individual observational units, but in these cases traditional one-at-a-time tests are severely lacking in power. In the population genetics case the goal is to identify genes under natural selection and additionally estimate the selection coefficients. Here we may have hundreds or thousands of genes. While each gene is assumed independent of the other genes, they all share the same phylogeny. This similar set of circumstances in the evolution of each gene means that much can be gained by assuming in our analysis that the responses come from some common distribution. In the proteomics application we are looking to identify which proteins have differential protein abundance across treatments. Here again, we may have hundreds or possibly thousands of proteins to analyze, and the similarity in circumstances across proteins indicates that combining infor-

mation across observations would improve estimation of individual effects.

The modeling frameworks presented here capitalize on the similarity of context of our observations to estimate "global parameters" with James-Stein type estimation of the subject specific effects. The fully Bayesian results discussed here were implemented with vague or "non-informative" priors. Particularly in the population genetics case, researchers may have strong *a priori* evidence they wish to incorporate into the analysis (eg, evidence of population divergence time or mutational constraint, etc) which is easily done in this context. Bayesian analysis, long avoided due to computational constraints, has become approachable for a much larger population with the development of software such as OpenBUGS and JAGS. With these now widely (and freely) available, lack of specific knowledge of MCMC samplers is no longer an impediment for fitting Bayesian models.

Empirical Bayes analysis, a term first coined by Robbins (1956), can be employed in situations like those described here "without even upsetting timid frequentists like myself" quips B. Efron (2008). In fact, the prevalence of data of this structure is such that "after 50 years of under use, we are poised for an avalanche of of empirical Bayes applications" (Efron, 2003). Thus, not so surprisingly we are currently working on extensions of the methods presented here to other applications.

As mentioned earlier, in addition to the structural similarity, both applications presented seek to "classify" the observations. In the SnIPRE methodology this is accomplished by construction of the confidence intervals about the random effects. This construction is straight-forward in the Bayesian setting where sampling from the posterior distribution is possible. In Chapter 3 we present a

new methodology based on profiled h-likelihoods for more precise estimation of the confidence bounds in the empirical Bayes setting.

The PROWLRE methodology, discussed in Chapter 5, takes a different approach to classification than SnIPRE, by incorporating a latent variable that indicates a protein's appropriate "class" into the model. The Bayesian framework is immediately amenable to such a model and this work is discussed explicitly in Booth et al. (2011). We focus here on the implementation in an empirical Bayes framework and propose an EM algorithm for fitting the model.

In the following chapter we introduce the SnIPRE methodology and compare it to similar methods currently available and in use via simulated data. Chapter 3 addresses the robustness of the application of SnIPRE to estimation of population genetics parameters when the assumption of independent sites is not met; introduces an alternative method for constructing confidence intervals for subject-specific effects in the empirical Bayes setting; and applies the methodology to *Drosophila* and human-chimp comparison data sets. Chapter 4 is a discussion of the SnIPRE methodology. Chapter 5 introduces the PROWLRE method and compares the empirical Bayes implementation to its fully Bayesian counterpart on synthetic and simulated data.

CHAPTER 2

SNIPRE: SELECTION INFERENCE USING POISSON RANDOM EFFECTS

In this chapter we present an approach for identifying genes under natural selection using polymorphism and divergence data from synonymous and non-synonymous sites within genes. A generalized linear mixed model is used to model the genome-wide variability among categories of mutations and estimate its functional consequence. We demonstrate how the model's estimated fixed and random effects can be used to identify genes under selection. The parameter estimates from our generalized linear model can be transformed to yield population genetic parameter estimates for quantities including the average selection coefficient for new mutations at a locus, the synonymous and non-synonymous mutation rates, and species divergence times. Furthermore, our approach incorporates stochastic variation due to the evolutionary process and can be fit using standard statistical software. The model is fit in both the empirical Bayes and Bayesian settings using the lme4 package in R, and Markov chain Monte Carlo methods in WinBUGS. Using simulated data we compare our method to existing approaches for detecting genes under selection: the McDonald-Kreitman test, and two versions of the Poisson random field based method MKprf. Overall, we find our method universally outperforms existing methods for detecting genes subject to selection using polymorphism and divergence data.

2.1 Background

Populations evolve over time and how they evolve is the product of different evolutionary forces. Population genetic theory gives us mathematical descriptions of how each of these forces is thought to affect the patterns of genetic variability within and between species. However, if the goal is not to start with an evolutionary model and see what happens, but rather to start with the data and understand what caused it one usually encounters an identifiability issue. For this reason, most population genetic data analyses looking for mutations under selection start by assuming a neutral population genetics model (constant population size, panmictic population, no migration), and test for deviations from this model. Commonly used examples of such procedures include tests based on summary statistics of the site frequency spectrum (distribution of mutation frequencies), such as Tajima's D (Tajima, 1989). However, since demographic factors (eg population growth) also effect the site frequency spectrum these tests are usually inconclusive. Tests based on linkage dis-equilibrium are also quite sensitive to demography as well as assumptions on recombination rates (Nielsen, 2005). The HKA statistic (Hudson et al., 1987) makes use of divergence data as well as within species variation by estimating the the variance of divergence to polymorphism ratios among loci. However, migration will result in a high variance of coalescent times among the loci, making the HKA test also sensitive to demography (Nielsen, 2001). See Nielsen (2005), for an excellent review of these procedures.

One class of tests which is robust to demography are those tests commonly referred to as "McDonald-Kreitman-type tests". This class includes the McDonald-Kreitman test (McDonald and Kreitman, 1991b) as well as MKprf (Bustamante

et al., 2003). The theory behind the McDonald-Kreitman test is developed in the following section.

Unlike many of the tests mentioned above, the method we present here assumes no particular population genetic model - in other words it is a non-parametric approach. Similar to the MK statistic, it is also robust to demography. Our method, which we call SnIPRE for *S*electio*n* *I*nference using *P*oisson *R*andom *E*ffects, works by modeling the variation within and between species as a combination of four types of "effects", one for each class of variation. These effects are functions of unknown population parameters of interest, including the selection coefficients.

Previously, we have developed a suite of powerful approaches that can estimate the average strength of selection operating on a locus and/or the distribution of fitness effects under specified population genetic setting for MK polymorphism and divergence data (see Bustamante et al. (2002); Barrier et al. (2003); Gilad et al. (2003); Bustamante et al. (2005); Sawyer et al. (2003); Boyko et al. (2008)). A main advantage of the "MKprf" approach is that it is much more powerful than carrying out individual MK tests and then correcting for multiple tests. A perceived disadvantage to some investigators is that it requires specifying a population genetic model and then fitting the parameters of that model. Some investigators have also been concerned about the use of Bayesian priors on the distribution of effects and the impact these can on inference (Li et al., 2008).

The main advantage of SnIPRE is that it can reliably identify genes under weak and strong negative as well as positive selection without needing to specify a population genetic model a priori. Nonetheless, because it "borrows in-

formation" from the rest of the genome regarding the average and variance in polymorphism to divergence, it outperforms the one-at-time MK test.

In this paper we will develop the model and the interpretation of its terms, and then describe how that model can be fit in both the empirical Bayes (SnIPRE) and fully Bayesian (B SnIPRE) settings. We also show how this model is robust to demographic history and recombination using standard coalescent simulations. Furthermore, we demonstrate how the Poisson Random Field estimates of average selection intensity, species-split time, mutation rate, and degree of selective constraint at the locus can be "extracted" directly from the SnIPRE estimates. We then compare the SnIPRE methods to the MK statistic and MKprf methods in detecting and estimating selection and other population parameters in simulations, and apply SnIPRE to data from a *Drosophila* and human-chimpanzee comparison data.

The MK statistic

Because SnIPRE works by picking up on the same type of signature of selection as the MK statistic, we will start with a review of this method and the theory behind it. While most techniques to identify loci under selection require assumptions about demography (particularly constant population size and no substructure), the MK statistic does not. Like the HKA statistic, it works by comparing divergence information between inferred neutral sites (such as synonymous sites in a protein-coding gene) and sites potentially under selection (such as non-synonymous sites at the same gene). The MK table consists of counts for four categories of mutations which occur in the coding region of a gene: polymorphic synonymous, divergent synonymous, polymorphic non-synonymous,

and divergent non-synonymous, see Table 2.1. It is important to note that the MK approach has also been extended to non-coding sites whereby upstream regions of a gene are compared to neighboring introns or synonymous sites (Andolfatto, 2005).

Table 2.1: **MK table.**

	Polymorphic	Divergent
Synonymous	y_{00}	y_{01}
Non-Synonymous	y_{10}	y_{11}

y_{ij} = the number of mutation a gene has in category ij; $i = 1$ if the mutations are non-synonymous, 0 otherwise; $j = 1$ if the mutations are divergent, 0 otherwise.

A mutation that occurs in every individual in the sample from one species is considered divergent, otherwise considered polymorphic. A mutation that occurs where it changes the amino acid produced is considered non-synonymous, otherwise considered synonymous. If the mutations are neutral, one would expect the ratio of polymorphic synonymous (PS) to divergent synonymous (DS) mutations to be the same as the ratio of polymorphic non-synonymous (PN) to divergent non-synonymous (DN) mutations, $PS/DS \approx PN/DN$. If this is not true, then we are seeing either an excess of DN mutations, or shortage DN mutations. An excess of DN is evidence supporting positive selection because mutations that change the amino acid are being fixed in the population at a higher rate red(but, also see Eyre-Walker (2002)). A shortage of DN is evidence of negative selection because mutations that change the amino acid are being fixed at a lower rate.

McDonald and Kreitman (McDonald and Kreitman, 1991a) use Fisher's exact test of independence on MK tables to identify genes under selection. This test can be justified using coalescent theory, where mutations are Poisson dis-

tributed across a gene genealogy G with expected value $\frac{\theta t}{2}$ across a segment of length t. Thus, conditioning on the total mutations (sufficient statistic for tree length) we have that $DS \mid PS + DS = n_1 \sim Bin(p_1, n_1)$ and $DN \mid PN + DN = n_2 \sim Bin(p_2, n_2)$. We wish to test $H_0 : p_1 = p_2$, the probability that a synonymous mutation becomes fixed is the same as the probability that a non-synonymous mutation appears fixed in the sample. Under this null hypothesis, $DN \mid DS + DN = d$ follows a hypergeometric distribution with parameters, $(n_1 + n_2, n_2, d)$.

$$P(X = DN \mid n_1 + n_2, n_2, d) = \frac{\binom{n_2}{DN}\binom{n1}{d-DN}}{\binom{n_1+n_2}{d}}$$

As long as the non-synonymous and synonymous sites are interspersed among each other, they will be similarly affected by demography and have the same distribution of coalescent times, thus the test is robust to demography.

Motivated by the MK statistic, the SnIPRE framework uses the MK table polymorphism and divergence data for identifying genes under selection. Using generalized linear mixed models we incorporate genome wide effects into our analysis as fixed effects, and individual gene effects as random effects. This method allows us to pool information across genes which increases our power to detect those under selection.

MKprf is another method that was developed by us which directly estimates the posterior distributions of genomic parameters, such as the species divergence time, based on the MK tables' synonymous cell entries. The posterior of the selection coefficients for each gene are then calculated conditional on these genomic parameters and the non-synonymous cell entries in the MK table, see Bustamante et al. (2002).

2.2 Methods

2.2.1 Data

The data consists of MK table counts for each gene, as well as the total number of synonymous sites and non-synonymous sites surveyed. Incorporating the number of sites into our model allows us to extend our inference beyond the interaction between non-synonymous and divergent mutations to include effects due to increases or decreases in the rate of non-synonymous mutations, and estimates of mutations rates and the times to the most recent common ancestor for each gene.

2.2.2 Model

Let K be number of genes in the sample. Thus we have $4K$ mutation counts y_{ijk}, where $i = 1$ if the mutation is non-synonymous, 0 otherwise, $j = 1$ if the mutation is fixed in the sample among the two populations being compared, 0 otherwise, and $k = 1, ..., K$ according to gene identification number. The mutation counts are assumed to be Poisson distributed, $y_{ijk} \sim P(\mu_{ijk})$, conditional on the covariates. The log of the expected mutation count is modeled using a generalized linear mixed effects model. The fixed effects include an intercept, an effect if the mutation is non-synonymous, an effect if the mutation is fixed, and an interaction effect if the mutation is both fixed and non-synonymous. Additionally the model includes four random effects: a gene effect, and the two-way and three-way interactions between the gene, non-synonymous, and divergence effects. Additionally, an offset term is used to control for the number

of sites sampled in the gene where a mutation of type i could occur, $Tsites_0$ for synonymous mutations, $Tsites_1$ for non-synonymous mutations.

$$\log(\mu_{ijk}) = \log(Tsites_i) + \beta + \beta^N i + \beta^D j + \beta^{ND} ij + \beta_k^G + \beta_k^{NG} i + \beta_k^{DG} j + \beta_k^{NDG} ij \quad (2.1)$$

Of primary interest is identifying genes under selection, either positive or negative. Identification of these genes can be done quite easily in the SnIPRE framework with only the assumptions of the MK test: i. synonymous and non-synonymous sites sampled are interspersed; ii. synonymous sites are not under selection. The non-synonymous-divergent interaction effects, β^{ND} and β^{NDG}, capture an average genome-wide selection effect and the gene-by-gene selection effects. The gene-by-gene selection effect β_k^{NDG} for a particular gene k, captures how the k^{th} gene varies from the average selection effect, β^{ND}, of all genes included in the sample. The k^{th} gene's selection effect relative to neutrality is reflected in the sum of these two interaction terms, $\beta^{ND} + \beta_k^{NDG}$. Thus, we refer to $\beta^{ND} + \beta_k^{NDG}$ as the *selection effect* for the k^{th} gene.

The other terms in the SnIPRE model are also quite interpretable. The intercept and the gene specific effect, β and β^G reflect the mutation rate. Here again the β_k^G term captures how the mutation rate for the k^{th} gene varies from the average mutation rate of the genes in the sample, β. We refer to $\beta + \beta_k^G$ as the *gene effect*. Similarly, β^D and β^{DG} reflect divergence time, and $\beta^D + \beta^{DG}$ is referred to as the *divergence effect*. The proportion of non-synonymous mutations that are non-lethal are reflected in β^N and β^{NG}. We refer to $\beta^N + \beta^{NG}$ as the *constraint effect*. These relationships are summarized in Table 2.2.

The model is fit in R (R Development Core Team, 2011) using the lme4 pack-

Table 2.2: **SnIPRE coefficients and population genetic parameters.**

Terms	Related parameters
$\beta + \beta_k^G$	θ_k, mutation rate for the k^{th} gene
$\beta_j^D + \beta_{jk}^{DG}$	τ_k, divergence time for the k^{th} gene
$\beta_i^N + \beta_{ik}^{NG}$	f_k, relative proportion of non-lethal non-synonymous mutations for k^{th} gene
	γ_k, selection coefficient for k^{th} gene
$\beta_{ij}^{DN} + \beta_{ijk}^{DNG}$	γ_k, selection coefficient for k^{th} gene
	τ_k, divergence time for k^{th} gene

Summary of the relationship between SnIPRE coefficients and population genetic parameters.

age (Bates et al., 2011), and a Bayesian GLMM is also fit using WinBUGS (Lunn et al. (2000), Sturtz et al. (2005)). In the Bayesian setting (B SnIPRE) we construct credible intervals for these effects based on the MCMC samples. In the empirical Bayes setting (SnIPRE) confidence intervals are constructed for the random effect estimates based on the standard errors. When fitting SnIPRE using the lme4 package we specified a general covariance structure. Allowing the general covariance structure, versus assuming the random effects are independent of each other, greatly improves the fit of the model and improves the prediction of genes under selection. Modeling a general covariance structure makes sense intuitively. For example, for a particular gene the non-synonymous and selection effects are especially likely to be correlated as selection affects the amount of time a non-synonymous mutation exists as a polymorphism before becoming fixed or eliminated. The selection effect reflects the selection coefficient γ, and the non-synonymous effect reflects mutation constraint, $1 - f$. Because of this relationship, one may be interested in examining the joint distribution for these estimated effects for a particular gene. This is easily accomplished in the Bayesian setting using the MCMC chains. As an example, see Figure 2.1.

For the Bayesian model the fixed effects have Normal priors with mean

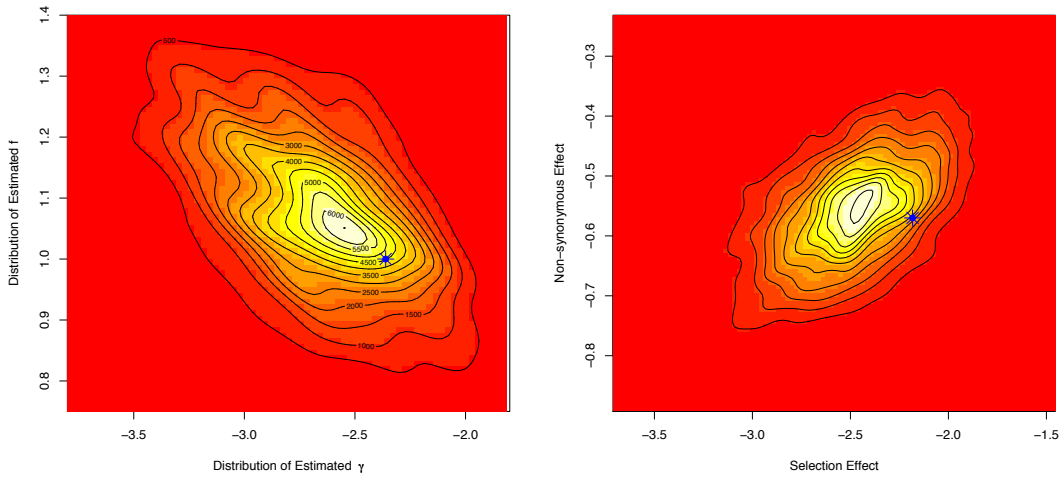


Figure 2.1: **Example joint distribution of the estimated selection effect and the constraint effect for a particular gene.** Data simulated using PRFREQ. The blue asterisk denotes the true location of parameters.

$\mu = 0$, and precision $\tau = .01$ priors. The priors for the random effects for each gene were multivariate normal with mean $\mu = (0, 0, 0, 0)$, and precision $\Psi_{4\times4}$. The precision matrix is modeled as a hyperparameter in order to estimate the covariance structure among the random effects. Using the conjugate prior, the Wishart distribution, we set $\Psi \sim W(S_{4\times4}, 10)$, where $S_{4\times4}$ is the identity matrix. Because the mutation counts are low these priors are considered non-informative.

An alternative formulation of the Bayesian model using hierarchical centering maybe be preferable as it results in quicker convergence (Gelfand et al., 1995). In the hierarchical centering formulation the fixed effects appear as hyper parameters about which the random effects are centered. The models are equivalent and as long as convergence criteria are met will yield the same inference.

2.2.3 Coalescent and Poisson random field frameworks

In standard coalescent theory we have j lineages coalescing at time points exponentially distributed with rate equal to $\frac{j(j-1)}{2}$. The number of segregating sites follows a Poisson process with rate $\theta/2$ per unit of time. Conditioning on the length of genealogy, t, which is a function of the coalescent times, the number of segregating sites is Poisson distributed with mean $\frac{\theta t}{2}$. Thus, we have the expected mutation count, μ_{ijk}, is a function of the sample coalescent times, as well as the mutation rate θ (Wakeley, 2007). Additionally, the expected mutation count should be adjusted for constraint, f, and selection γ. This is consistent with our model where the effects of mutation rate and divergence is estimated from the synonymous mutations, and constraint and selection are estimated from the non-synonymous.

Our model also works well in the Poisson random field (PRF) framework which assumes i. mutations arise at exponentially distributed times, ii. each mutation occurs at a new site, and iii. each mutant follows and independent Wright-Fisher process (i.e. no linkage) (Sawyer and Hartl, 1992). SnIPRE can be viewed as a re-parameterization of the PRF framework. Thus it is convenient to use the relationships between the SnIPRE coefficients and and the PRF model to obtain estimates of γ ($\gamma = 2N_e s$, where 1+s is the fitness of mutants, and N_e is the effective population size), as well as τ, f, and θ ($\theta = 4N_e u$ where u is the nucleotide mutation rate, and N_e is the effective population size). We can derive the relationship between the population genetic parameters and the SnIPRE coefficients by comparing the predicted MK table counts provided by SnIPRE, see Table 2.3, which are written in terms of model coefficients, to the theoretical expected MK table counts given in Table 2.4. These relationships are

derived below; n and m represent the number of samples from the population of interest and the out-group.

Table 2.3: **SnIPRE predicted mutation counts.**

	Polymorphic	Divergent
Syn	$T sites_0 \exp(\beta + \beta^G)$	$T sites_0 \exp(\beta + \beta^G + \beta^D + \beta^{DG})$
Non-syn	$T sites_1 \exp(\beta + \beta^G + \beta^N + \beta^{NG})$	$T sites_1 \exp(\beta + \beta^G + \beta^N + \beta^{NG} + \beta^D + \beta^{DG} + \beta^{ND} + \beta^{NDG})$

The predicted mutation counts expressed in terms of the number of synonymous and non-synonymous sites sampled $T sites_0$, $T sites_1$, the gene effect β^G, non-synonymous effect β^N, divergent effect β^D, and their interactions.

Table 2.4: **Expected mutation counts.**

	Polymorphic	Divergent
Syn	$T sites_0 \theta [L(m) + L(n)]$	$T sites_0 \theta \left(\tau + \frac{1}{m} + \frac{1}{n}\right)$
Non-syn	$T sites_1 f \theta \frac{2\gamma}{1-e^{-2\gamma}} [F(m) + F(n)]$	$T sites_1 f \theta \frac{2\gamma}{1-e^{-2\gamma}} [\tau + G(m) + G(n)]$

$$L(n) = \sum_{i=1}^{n-1} \frac{1}{i} \tag{2.2}$$

$$F(n) = \int_0^1 \frac{1 - x^n - (1-x)^n}{1-x} \frac{1 - e^{-2\gamma x}}{2\gamma x} dx \tag{2.3}$$

$$G(n) = \int_0^1 (1-x)^{n-1} \frac{1 - e^{-2\gamma x}}{2\gamma x} dx \tag{2.4}$$

The expected mutation counts expressed in terms of the number of synonymous and non-synonymous sites sampled $T sites_0$, $T sites_1$, selection coefficient γ, the species-split time τ, the mutation rate θ, the proportion of lethal non-lethal mutations f, and the number of samples in the population of interest and the out-group n and m, according to the Poisson Random Field framework.

The gene effect $\beta + \beta^G$, is a function of the mutation rate θ.

$$\exp(\beta + \beta^G) = \theta [L(m) + L(n)] \tag{2.5}$$

The divergence effect, $\beta^D + \beta^{DG}$, is a function of the divergence time τ.

$$\exp(\beta^D + \beta^{DG}) = \frac{\exp(\beta + \beta^G + \beta^D + \beta^{DG})}{\exp(\beta^+ \beta^G)} \tag{2.6}$$

$$= \frac{\left(\tau + \frac{1}{m} + \frac{1}{n}\right)}{[L(m) + L(n)]} \tag{2.7}$$

The selection effect $\beta^{ND} + \beta^{NDG}$, is a function of the selection coefficient γ, and the time to the most recent common ancestor τ. The selection effect, $\beta^{ND} + \beta^{NDG}$, reflects the interaction effect between the mutations in a gene being both non-synonymous and divergent on the mutation count. A positive selection effect indicates that mutations that are non-synonymous are being fixed at a higher rate than expected under the null hypothesis of no selection. A negative selection effect indicates that mutations that are non-synonymous are being fixed at a slower rate than expected. Assuming a neutral demography, a positive (negative) selection effect corresponds to a positive (negative) selection coefficient. That positive (negative) selection leads to the higher (lower) rate of fixation for non-synonymous mutations makes sense intuitively.

$$\exp(\beta^{ND} + \beta^{NDG}) = \frac{\exp(\beta + \beta^G + \beta^N + \beta^{NG} + \beta^D + \beta^{DG} + \beta^{ND} + \beta^{NDG})}{\exp(\beta + \beta^G + \beta^N + \beta^{NG}) \exp(\beta + \beta^G + \beta^D + \beta^{DG})} \times \exp(\beta + \beta^G) \tag{2.8}$$

$$= \frac{[\tau + G(m) + G(n)]\,[L(m) + L(n)]}{[F(m) + F(n)]\left(\tau + \frac{1}{m} + \frac{1}{n}\right)} \tag{2.9}$$

The non-synonymous effect, $\beta^N + \beta^{NG}$, is a function of the proportion of non-lethal non-synonymous mutations relative to synonymous mutations, f, as well as the selection coefficient, γ. The constraint effect, $\beta^N + \beta^{NG}$, reflects the effect that mutations being non-synonymous (versus synonymous) has on the expected count. A negative (positive) constraint effect indicates that non-synonymous

polymorphic mutations are either being fixed or eliminated at a higher (lower) rate than synonymous mutations. Thus, after estimating the selection coefficient to account for the rate at which non-synonymous mutations are fixed, we can estimate from the constraint effect the proportion of mutations that are lethal, and therefore quickly eliminated from the population.

$$\exp(\beta^N + \beta^{NG}) = \frac{\exp(\beta + \beta^G + \beta^N + \beta^{NG})}{\exp(\beta + \beta^G)} \tag{2.10}$$

$$= \frac{f\frac{2\gamma}{1-e^{-2\gamma}}\left[F(m) + F(n)\right]}{\left[L(m) + L(n)\right]} \tag{2.11}$$

It is interesting to note that these are the relationships used by Sawyer and Hartl (1992) to fit their single locus PRF models to 2×2 MK data. What is different about our approach is that we do not require a PRF parameterization for inference; rather, it naturally falls out from consideration of the standard log-linear model analysis of multi-way contingency tables. Several of the simulations in the next section are done in the PRF framework using PRFREQ (Boyko et al., 2008). We have also done several simulations using SFS_CODE (Hernandez, 2008) that show our estimation of population genetic parameters to be fairly robust to the PRF assumption of no linkage between sites. Specifically, the false positive rate remains low for identification of genes under selection. The primary consequence of linkage is underestimation of the magnitude of selection. We plan to explore these results more in a later paper. The simulations found in Table 2.6 were all conducted using SFS_CODE and include recombination.

2.3 Simulations and Discussion

To assess and compare the performance of the SnIPRE methods against the MK statistic and MKprf method we simulated data using 3 different methods. The first method, based on coalescent theory, was implemented in R. The second method, PRFREQ, simulates data based on the PRF framework. The third method is a forward simulation method, SFS_CODE. In these simulations our first goal was to compare the false positive rates of the methods using simulations under neutrality. Additionally, we simulated data with selective constraint but without selection which illustrates SnIPRE's ability to distinguish between mutational constraint and selection. Using PRFREQ, we were also able to simulate data sets with a distribution of selection coefficients and use this to compare the methods in a litany of non-neutral settings.

2.3.1 Simulations under neutrality

To assess false positive rate FPR for each of the methods, we simulated data using standard coalescent theory. In Table 2.5, we report the false positive rate for a data set with 1,000 neutrally evolving genes simulated from a pair of populations of constant size that split $\tau = 10 \times 2N_e$ generations ago, with mutation rate $\theta = 4N_e u = .001$. The standard MK approach had an FPR = 0.02. SnIPRE performed very well with an FPR< 0.001 for both the Bayesian and empirical Bayes approaches. MKprf had mixed performance, depending on assumptions regarding the variance of the distribution of fitness effects. For fixed variance, $\sigma^2 = 10$, the FPR = 0.14 which is relatively high. This is a mode of MKprf that has a very wide prior distribution that is not updated by information from other

loci. When that information is incorporated we see that MKprf (estimated σ^2) also has a low FPR, 0.012.

Table 2.5: **False positive rate.**

Method	False Positive Rate
SnIPRE	0.00
B SnIPRE	0.00
MKprf ($\sigma^2 = 10$)	0.14
MKprf (estimated σ^2)	0.01
MK	0.02

False positive rate in a data set of 1000 genes simulated using the coalescent method.

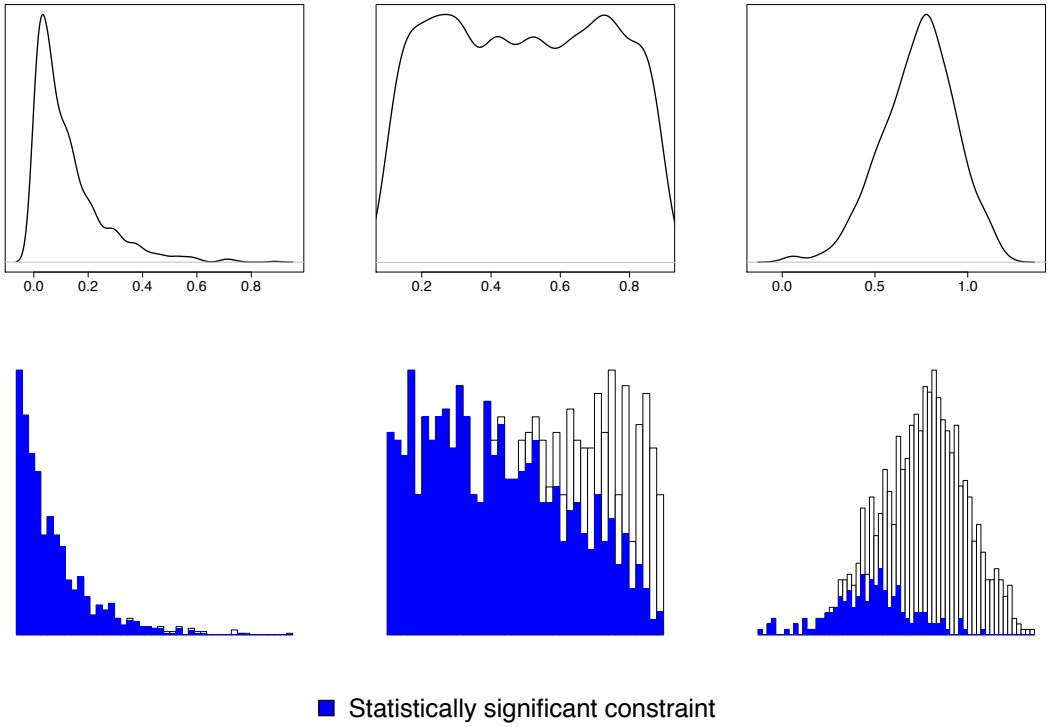


Figure 2.2: **Classification of constraint.** Top: Distribution 1, 2, and 3 of f used in the coalescent simulations for Table 2.7. Bottom: Shaded regions of the histogram represent the proportion of constraint effects classified as significant by SnIPRE; constraint effects binned by true value of f.

Next we investigated the impact of demographic history as well as recombination on the FPR of the methods using the forward simulator SFS_CODE. In Table 2.6, we report simulation results for 5 demographic settings for 1,000 gene data sets including three bottleneck scenarios, one population growth model, and constant population size. From these simulations we see that both the MK method and SnIPRE methods have very low false positive rates, with the SnIPRE method performing slightly better. MKprf with estimated variance has similarly very low false positive rates, however MKprf with $\sigma^2 = 10$ has consistently higher false positive rates. As stated above, all these methods should be robust to demography. This appears to be the case in our simulations as the false positive rates remains consistent for each method across demographies.

Table 2.6: **False positive rate and demography.**

	Bottleneck 1	Bottleneck 2	Bottleneck 3	Expansion	Constant
SnIPRE	0.00	0.00	0.00	0.00	0.00
B SnIPRE	0.00	0.00	0.00	0.00	0.00
MKprf ($\sigma^2 = 10$)	0.11	0.08	0.01	0.11	0.13
MKprf (estimated σ^2)	0.00	0.00	0.00	0.00	0.03
MK	0.02	0.02	0.01	0.02	0.03

False positive rates when no selection, and under various population growth models.

The key point from all these simulations is that SnIPRE performs just as conservatively as the MK test and better than MKprf under a litany of neutral scenarios that might be cause for concern in analyses for inference of selection.

2.3.2 Simulations with constraint

A particularly interesting application of SnIPRE is to identify regions of the human (or a new genome) that show very low levels of variation based on both

polymorphism and divergence data. These might be interpretable as regions of high selective constraint either at the amino acid or non-coding level (for comparison with a flanking "neutral" standard) and may represent biologically meaningful sequences, see Pollard et al. (2006), and Bejerano et al. (2006).

To quantify the power of SnIPRE to identify constrained loci, we simulated three different scenarios with varying degree of selective constraint, or f, among genes in 1,000 gene data sets. Here we consider the case were some proportion of sites are very strongly constrained (any mutation at these locations is considered lethal), and not the case where the mutations are of weak negative effect and could rise in frequency and contribute to polymorphism (considered in the simulations below). That is, these regions do not exhibit a deviation in polymorphism vs. divergence; however, they will be outliers with regard to the genome-wide pattern of overall genetic variation. In Table 2.7 and Figure 2.2 we see the results from three coalescent simulations with three different distributions on selective constraint, f. A comparable estimate of constraint from the MKprf methods is a function of its estimated non-synonymous and synonymous mutation rates θ_N, and θ_S:

$$\frac{\theta_N / \#\text{ Non-Synonyomous sites}}{\theta_S / \#\text{ Synonymous sites}}$$

The SnIPRE methods performed quite well on data from distribution one with 98% and 99% correct, the MKprf methods yielded only 43% and 67% correct. Distribution 2 has a wider variety of constraint and presents more of a challenge for both SnIPRE (66% and 86%)and MKprf (38% and 51%) methods. Distribution three contained only mild to moderate constraint and was the most challenging of the three distributions. Here, the B SnIPRE method proved to be the most powerful of the four methods, with 45% correctly classified, and the MKprf methods yielded approximately 21% correct, and SnIPRE approximately

17% correct. For all three distributions the SnIPRE methods correctly classified the selection effects as neutral. From these results we see that the SnIPRE model is able to detect strong constraint, and can distinguish these effects from those of selection.

Table 2.7: **SnIPRE results for simulations with a distribution on constraint.**

	% Correct γ			% Correct f		
	Dist 1	Dist 2	Dist 3	Dist 1	Dist 2	Dist 3
SnIPRE	100.0	100.0	100.0	98.7	66.1	17.9
B SnIPRE	100.0	100.0	100.0	99.2	86.0	43.4
MKprf ($\sigma^2 = 10$)	69.3	92.2	87.9	43.0	38.7	21.5
MKprf (estimated σ^2)	71.1	99.3	99.3	67.6	51.6	20.7
MK	99.2	98.2	97.4			

SnIPRE results for coalescent model simulations with a distribution on f, and no selection ($\gamma = 0$).

A comparison can also be made when selection is present, and there is no constraint ($f = 1$). To do this we considered a data set with selection coefficients drawn from a normal distribution with a mean of zero, and a standard deviation of two (see Distribution 2 in Figure 2.4), and with no constraint. In Figure 2.3 A we see that SnIPRE's estimated constraint effects are quite accurate (very close to one), while the MKprf methods have much more variable estimates. The SnIPRE method's estimates of constraint are somewhat correlated with the selection coefficient, however we see in Figure 2.3 B that the effect of this trend is minimal.

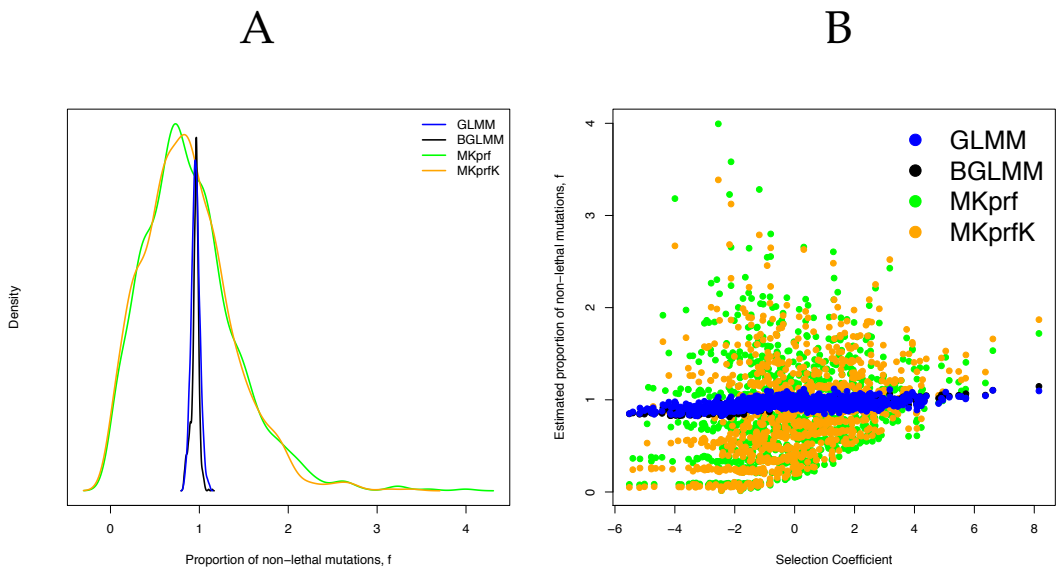


Figure 2.3: **Comparison of estimates of constraint when $f = 1$ (no constraint).** A: The distribution of constraint estimates. B: Constraint estimates versus the selection strength.

2.3.3 Simulations with selection

Classification of selection effect

To assess performance when the selection coefficients come from some distribution, we simulated data using PRFREQ for six data sets of 1000 genes. Selection coefficients for our simulations are drawn from three distributions, which are shown at the bottom of both Figure 2.4 and Figure 2.5. These selection coefficients were then used to simulate data with drosophila-like parameters $\theta = \rho = 0.01$, and with human-like parameters $\theta = \rho = 0.001$. In Figures 2.4 and 2.5 each row of histograms illustrates a particular method's performance on data from each of the simulations. The colored portions of the histograms represent the proportion of selection coefficients in each bin correctly classified as under selection, with the true selection coefficient values given along the x-

axis. These results are also summarized in Table 2.8. From our simulations, we found the SnIPRE method to be a dramatic improvement over other methods in identifying genes under selection, especially when table counts are low, as with a human-like mutation rate of $\theta = .001$. For example, the SnIPRE methods classify 72% – 88% of genes correctly, MKprf methods classify 42% – 60% correctly, and the MK statistic just 12% – 20% correctly. For the drosophila-like simulations the SnIPRE methods classify 90% – 95% correctly, MKprf methods classify 83% – 90% correctly, and the MK statistics classifies 67% – 77% correctly. Specifically, the SnIPRE methods are more sensitive for small (close to zero) and more accurate for extreme valued selection coefficients. The selection coefficients not identified by SnIPRE as significantly different from zero, are generally within ±1 of zero.

Table 2.8: **Selection classification for simulations by method.**

	$\theta = .01$ (Drosophila)			$\theta = .001$ (Human)		
	Dist 1	Dist 2	Dist 3	Dist 1	Dist 2	Dist 3
SnIPRE	0.95	0.92	0.95	0.86	0.72	0.88
B SnIPRE	0.93	0.90	0.93	0.85	0.76	0.86
MKprf ($\sigma^2 = 10$)	0.90	0.83	0.89	0.50	0.45	0.60
MKprf (estimated σ^2)	0.90	0.83	0.89	0.52	0.42	0.57
MK	0.77	0.67	0.76	0.20	0.12	0.15

Proportion of genes correctly classified under selection where the selection coefficients are from distribution 1, 2 and 3; mutation rate θ.

The methods were also tested on a data set which contained both genes with and genes without mutations under selection ($\theta = 0.001$, selection strength of at least ±1, simulation done in PRFREQ). In Figure 2.6 the true positive rate is plotted versus the false positive rate. Here we see that at the cutoff needed for the MK statistic to have identified half the genes under selection (TPR = 0.5), approximately half of the discoveries are false (FDR $\approx$ 0.5). The MKprf methods offers a dramatic improvement of the MK statistic with a FDR approximately

equal to 0.1 at a TPR = 0.5, but the SnIPRE methods offer further improvement with a FDR of zero at TPR = 0.5.

Estimation of selection coefficient, γ

As previously mentioned, the SnIPRE method can be used not only to classify selection effects as negative, neutral, or positive, but can also be used (with a few additional assumptions) to provide estimates of the selection coefficient, γ. We compare the SnIPRE and MKprf estimates of γ for the PRFREQ simulation data in Figures 2.4 and 2.5. The distribution of the differences between the estimates and the truth selection coefficient, $\hat{\gamma}_i - \gamma_i$, for each method by data set is shown in Figure 2.7. The SnIPRE methods generally yield reasonable results for genes with selection coefficients from -2 or higher. However for genes under strong negative selection cell counts are often quite small or zero, and since the cell counts are bounded below by zero it is hard to estimate precisely the extent of negative selection. Because of this, both the SnIPRE methods and MKprf method suffer in precise estimation of negative selection coefficients. However, as seen in Figure 2.5 the SnIPRE methods still classify these coefficients as negative, whereas MKprf does so for only a fraction of the more extreme cases.

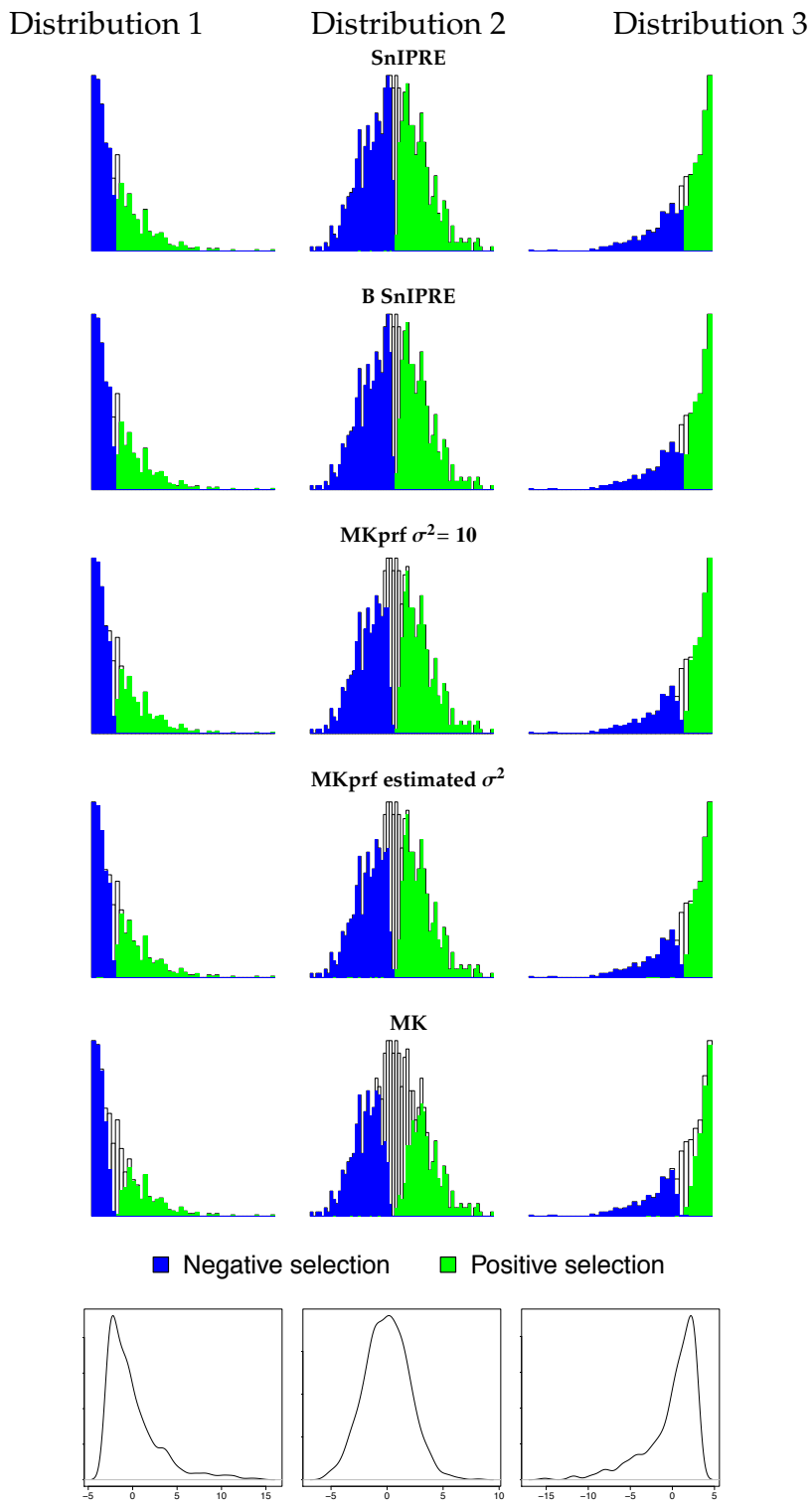


Figure 2.4: **Classification of selection effect for *Drosophila*-like simulations.** Shaded regions of histogram represent the proportion of genes under selection classified as under selection; x-axis is true selection coefficient; $\theta = 0.01$.

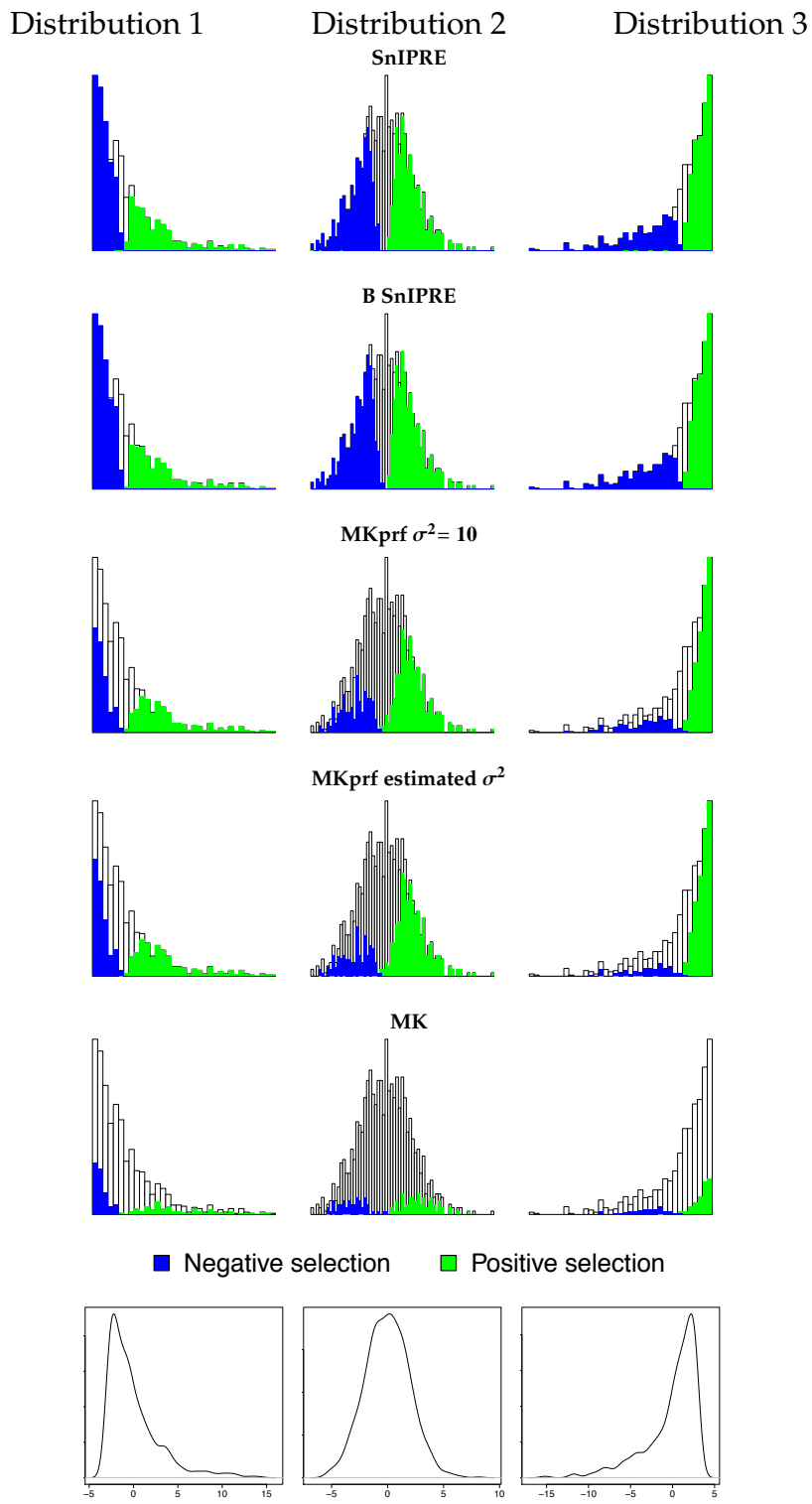


Figure 2.5: **Classification of selection effect for human-like simulations.** Shaded regions of histogram represent the proportion of genes under selection classified as under selection; x-axis is true selection coefficient; $\theta = 0.001$.

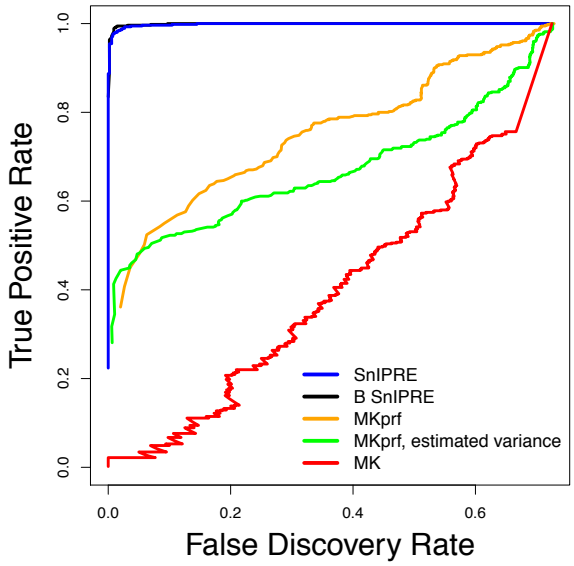


Figure 2.6: **True positive rate versus false discover rate.** Results for data set of 2000 genes, 550 of the genes are under selection with $\gamma < -1$ or $\gamma > 1$.

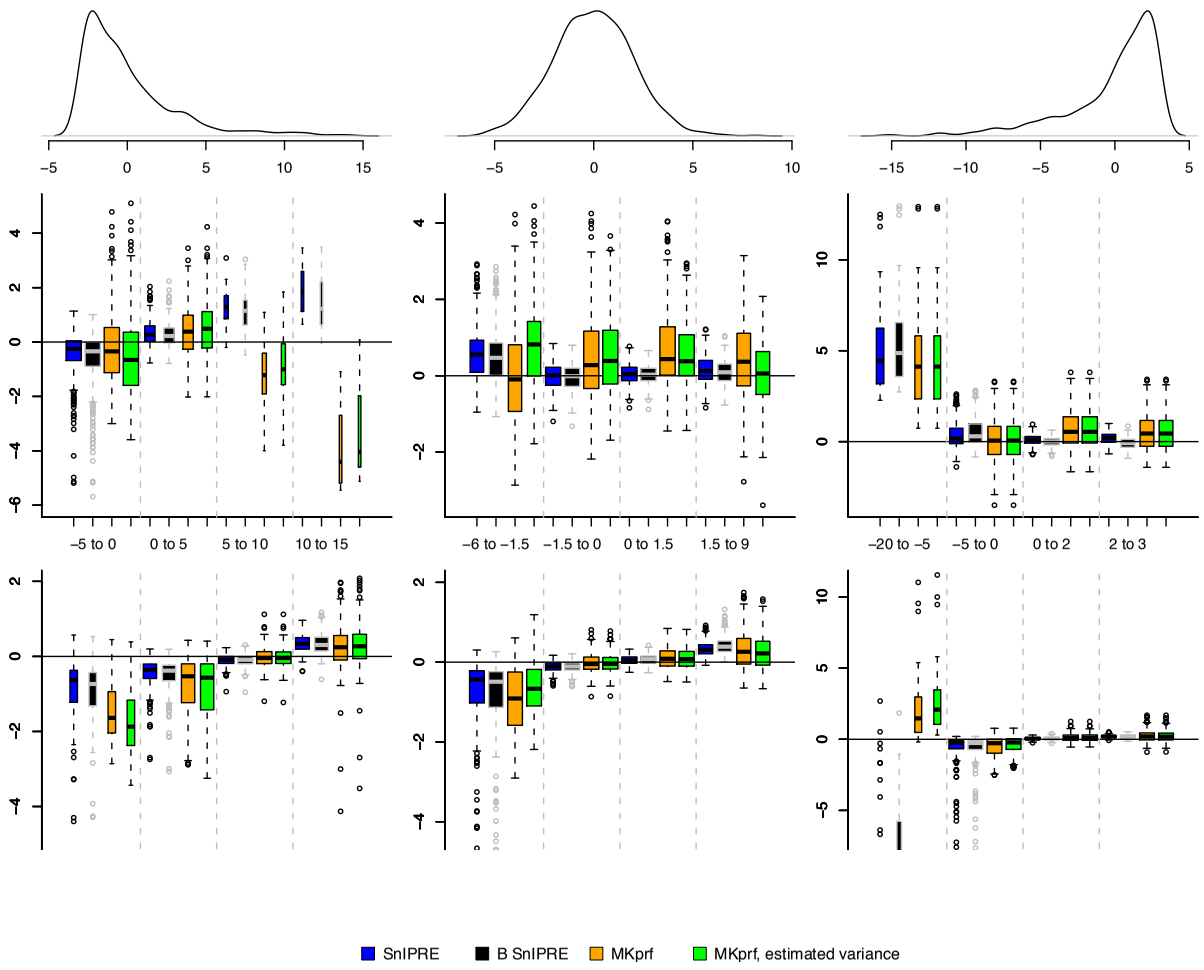


Figure 2.7: **Distribution of residuals for selection coefficient estimates by method.** Residuals grouped by true selection strength.

2.4 Conclusions

The SnIPRE framework models MK table data in a way consistent with population genetic theory and with minimal assumptions on the demographic model. The estimated effects of the model are easily interpreted and can be effectively used to estimate the affects of selection, constraint, divergence time, and mutation rate on genome-wide patterns of variation on a gene-by-gene basis.

The simulations provided here illustrate the significant increase in power over the traditional MK test that the SnIPRE model provides, while maintaining a low false positive rate. This makes sense since we are using genome-wide data to improve our estimate of the influence of mutation rate, species divergence time, constraint, and selection effects. The fixed effects reflect genome-wide averages of these effects; the random effects reflect the gene-by-gene variation in the influence of these forces and provide estimates of this variation with James-Stein-type shrinkage. Both the empirical Bayes and fully Bayesian implementation borrow strength across genes to improve estimates of the parameters of interest.

When the assumptions of the PRF are met, our simulations indicate the method provides estimates of the selection coefficient as un-biased as the more parametric method MKprf, and with generally smaller confidence intervals.

In the next chapter we explore the impact of varying recombination rate on the accuracy of parameter estimates and introduce an alternative method for calculating the confidence bounds for the selection effect in the empirical Bayes setting. We also will apply our method to *Drosophila* and human-chimpanzee comparison data.

CHAPTER 3

SNIPRE: PRACTICAL CONSIDERATIONS

The SnIPRE methodology is a powerful tool for identifying genes under selection. As shown in chapter 1, using SnIPRE to classify genes under selection is as straight-forward extension MK statistic. SnIPRE can also be used to quantify the strength and direction of selection, but this requires an additional assumption of free recombination (i.e. independent sites). We explore the robustness of the model to this assumption in section 2.1.

The previous chapter introduced both an empirical Bayes and fully Bayesian implementation of SnIPRE. The two versions yielded very similar results, however discrepancies can arise due to differences in the credible and confidence intervals as well as the notoriously difficult estimation of the random effects variance components. The Bayesian credible intervals are constructed to capture the middle 95%, however the empirical Bayes confidence intervals or prediction intervals are estimated by ±1.96 standard errors, necessarily constraining the intervals to be symmetric about the estimate. In section 2.2 we introduce a more accurate method of constructing confidence intervals for the random effects in the empirical Bayes framework using the Laplace approximation (De Bruijn, 1981).

In the section 2.3 we apply the SnIPRE methods to a *Drosophila simulans - Drosophila melanogaster* comparison data and *Homo sapiens-Pan troglodytes*, or human-chimpanzee comparison data.

3.1 Recombination

Recombination acts as a major engine of genetic variation by introducing new combinations of genotypes into the population. When mutations arise which are advantageous, natural selection will drive the mutation to fixation within a population. However, if separate individuals carry competing advantageous alleles this will increase the time until one allele becomes fixed (Hill-Robertson or Fisher-Muller effect (Hartl and Clark, 2006)). In the presence of recombination, both mutations may be passed on simultaneously and become fixed. The higher the rate of recombination, the more efficient the process of including both mutations on the same genotype, and the closer the process becomes to our model. In the presence of low recombination, we would expect the competing selection forces to result in an underestimation of the selection effect. To test this we simulated several data sets (each 500 genes) using the forward simulator SFS_CODE Hernandez (2008) and specified a range of recombination rates from $\rho = 0.001$ to $\rho = 0.9$ ($\rho = 4N_e r$ is the population scaled rate of cross-over between adjacent sites for a diploid population). In these simulations we used a human-like mutation rate of $\theta = .001$ ($\theta = 4N_e\mu$, the per-site population scaled mutation rate), and every new non-synonymous mutation has a selection coefficient of $\gamma = 5$ ($\gamma = 2N_e s$). As expected, as the recombination rate increases, our estimate of the average selection coefficient becomes more accurate see Figure 3.1. For low recombination rates, our method yields conservative estimates of the average strength of positive selection.

The Hill-Robertson effect affects not only our estimate of selection strength, but also that of constraint. Because adaptive mutations remain as polymorphisms in the population for extended periods of time when there are compet-

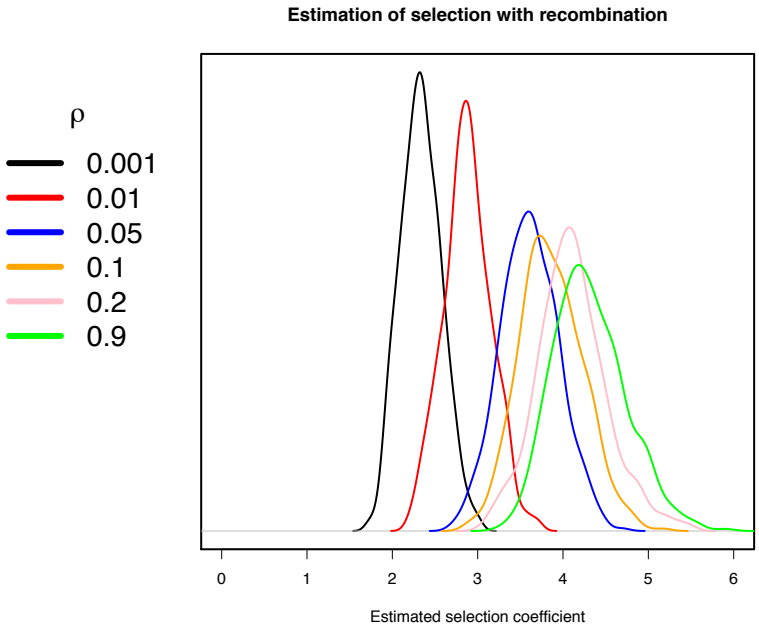


Figure 3.1: **Illustration of the effect of recombination on the estimate of selection, γ.** Here every mutation was evolutionarily advantageous ($\gamma = 5$, $\theta = .001$), however due to interference from competing alleles mutations are becoming fixed at a slower rate than expected under the PRF model leading to under-estimation of the selection coefficient.

ing advantageous alleles, there is an excess of non-synonymous polymorphism. This leads to an increase in the estimate of f, the proportion on non-lethal non-synonymous mutations. Thus, higher than expected $\hat{f}$, especially $\hat{f} > 1$ may be evidence of rampant positive selection when the recombination rate is low. We observed this to be true in our simulations. Figure 3.2 shows the estimated constraint levels for the same data examined in Figure 3.1. Here we see that those simulations with the lowest recombination have the highest $\hat{f}$. In these simulations, there was no constraint ($f = 1$). Recombination rates of $0.9, 0.2, 0.1$, and even 0.05 had unbiased estimates f. The lower rates (and more biologically realistic rates) of $\rho = 0.01$ and $\rho = 0.001$ lead to consistant over-estimation of

constraint when in the presence of rampant positive selection. (While the estimates of f are biased, they are not statistically significantly different from the true $f = 1$. All the genes were also correctly classified as under positive selection.)

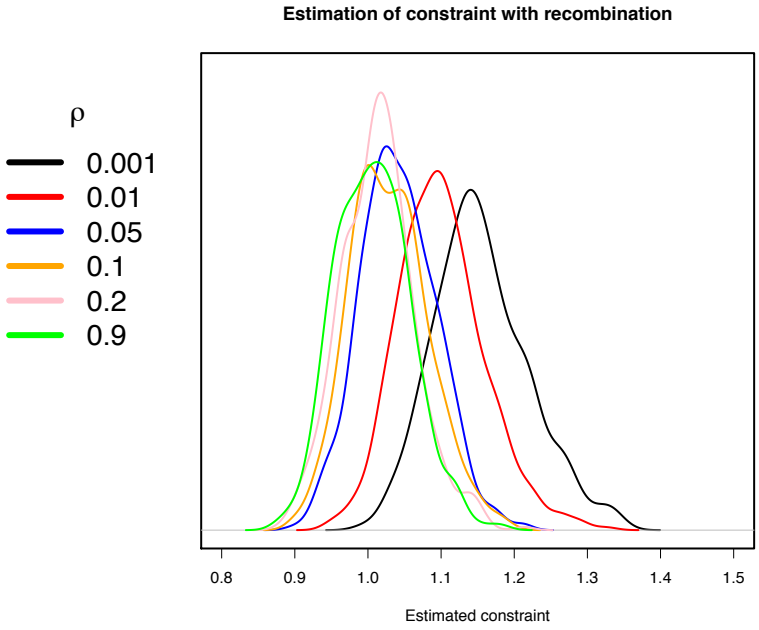


Figure 3.2: **Illustration of the effect of recombination on the estimate of constraint, f.** Due to interference from competing advantageous alleles mutations are becoming fixed at a slower rate than expected under the PRF model, leading to biased estimates of f for the lower recombination rates (true $f = 1$).

These results imply that in instances of rampant positive selection and low recombination, both the constraint and selection effects may be affected and should be considered together. Examining the joint density plots for genes of interest as discussed in Chapter 1 (see Figure 2.1) may be particularly useful.

3.2 Laplace approximation and confidence interval construction

The standard method for constructing confidence intervals for random effects (sometimes referred to as "prediction intervals") is based on their standard errors. For the linear mixed model

$$y = X\beta + Zu + e$$

with

$$e \sim MVN(0, W^{-1})$$

and

$$u \sim MVN(0, D)$$

we have

$$\begin{aligned} \hat{u} \equiv \mathrm{E}(u|y) &= (Z^T W Z + D^{-1})^{-1} Z^T W (y - X\beta) \\ \mathrm{var}(u|y) &= (Z^T W Z + D^{-1})^{-1} \\ &= \mathrm{var}(\hat{u} - u). \end{aligned}$$

The standard error for an individual random effect is therefor

$$\mathrm{se}(\hat{u}_i - u_i) = \sqrt{(Z^T W Z + D^{-1})^{-1}_{i,i}}, \tag{3.1}$$

and the construction of the confidence interval is the usual

$$\hat{u} \pm z_{\alpha/2} se(\hat{u}_i - u).$$

For linear mixed models, this formulation for the variance of $u \mid y$ is exact (Lee et al., 2006). For the generalized linear mixed model we can use the approximating linear mixed model

$$z = X\beta + Zu + e$$

where z is a working response. In the case of the SnIPRE model, we have poisson counts, so our working response is

$$z = \eta + (y - \mu)\frac{\partial \eta}{\partial \mu} = \log(\mu) + \frac{y - \mu}{\mu},$$

the design matrix Z will be $4nx4n$ (n = the number of genes). The residuals, e is the working residual, defined in our case as

$$e = \frac{(y - \mu)}{\mu}$$

and with precision

$$W = \text{diag}(\mu).$$

In this case $(Z^T WZ + D^{-1})^{-1}$ corresponds the variance estimate derived from the Laplace approximation to the likelihood function where β and D are fixed.

By construction these confidence intervals are symmetric. We will refer to the confidence intervals constructed in this fashion as LAMP confidence intervals for Laplace Approximated Marginal Posterior. We introduce an alternative method which approximates the marginal posterior by repeated implementation of the Laplace approximation and numerical quadrature for more precise estimation of a central 95% confidence interval.

3.2.1 Method

To obtain confidence intervals for the selection effects comparable to the Bayesian credible intervals ideally we would construct the intervals based on the marginal posterior of β^{NDG}. Letting the subject specific random effects for the k^{th} protein be represented by the vector u_k

$$[\beta^{G_k}\ \beta^{NG_k}\ \beta^{DG_k}\ \beta^{NDG_k}] = [u_{1k}\ u_{2k}\ u_{3k}\ u_{4k}] = u_k$$

and treating the global parameters β, β^N, β^D, β^{ND}, and Σ as fixed we can write the marginal posterior of interest as

$$f(\cdot \mid y) = \int \left\{ \prod_{i,j} \frac{\mu_{ijk}^{y_{ijk}}}{y_{ijk}!} e^{-\mu_{ijk}} \right\} \mid 2\pi\Sigma \mid^{-1/2} \; e^{-\frac{1}{2} u_k' \Sigma^{-1} u_k} du_{[1-3]k}. \tag{3.2}$$

However, this integral is analytically intractable. An approximation can be achieved through the application of the Laplace approximation.

Laplace Approximation

Generally, for a smooth, unimodal, and positive function we can write the integral of interest in the form

$$\int e^{h(x)} dx$$

for some function h. Find the parameter value x_0 such that h has a global max at $h(x_0)$. Now, using the Taylor expansion we have

$$h(x) = h(x_0) + h'(x_0)(x - x_0) + \tfrac{1}{2}(x - x_0)^T h''(x_0)(x - x_0) + \sum O(\mid x - x_0 \mid^3)$$

$$h(x) \approx h(x_0) \qquad\qquad + \tfrac{1}{2}(x - x_0)^T h''(x_0)(x - x_0)$$

The first derivative term disappears at the global max, so we can re-write our integral

$$\int e^{h(x)} dx \;\approx\; e^{h(x_0)} \int e^{\frac{1}{2}(x-x_0)^T h''(x_0)(x-x_0)} dx \tag{3.3}$$

$$= \; e^{h(x_0)} 2\pi^{k/2} \left| \left(\frac{-\partial^2 h}{\partial x \partial x'} \right) \right|^{-1/2}. \tag{3.4}$$

In the right hand side of equation 2.3 we see that the integrand is a an unscaled normal distribution with mean x_0 and variance equal to the inverse of the Hessian, resulting in equation 2.4.

For the SnIPRE model, from 3.2 the log-likelihood of interest is

$$h(u) = \log f(u \mid y) = \sum_i \sum_j (y_{ij} \log(\mu_{ij}) - \mu_{ij}) - \frac{1}{2} u' \Sigma^{-1} u$$

and we are integrating with respect to the first three elements of u. To estimate the marginal posterior of u_4 we need to evaluate 3.2 at several points covering the range of (likely) values for u_4. The algorithm is as follows.

Procedure

1. Pick a value for u_4, call it u_4^i, at which you would like the know the MP density
2. Find x_0^i s.t. global max at $h(x_0^i, u_4^i)$
3. Evaluate $h''(x_0^i, u_4^i)$ (See appendix for Hessian??).
4. Set $f(u_4^i \mid y) = e^{h(x_0^i, u_4^i)} \left| \frac{1}{2\pi} \left(\frac{-\partial^2 h(x_0^i, u_4^i)}{\partial u_{1-3} \partial u'_{1-3}} \right) \right|^{-1/2}$
5. Repeat steps 1 - 4 for different u_4^i until you have evaluated the MP at all points of interest
6. Use numerical quadrature to determine cutoff points for lower 2.5% and upper 2.5% confidence bounds

With repeated implementation of the Laplace approximation as outline above, the approximate marginal likelihood function is defined as

$$f(u_4^i \mid y) \approx \exp\{h(\hat{u}_{1-3}^i, u_4^i)\} \left| \frac{1}{2\pi} - h''(\hat{u}_{1-3}^i, u_4^i) \right|^{1/2} \tag{3.5}$$

where $\hat{u}_{1-3}^i$ are the MLEs conditional on $u_4 = u_4^i$. This approximation could be referred to as an adjusted profile h-likelihood, since $e^{h(x_0^i, u^i)} = \max_x L(x, u_4)$ is the profiled h-likelihood. This is similar to adjusted profile likelihood described by

Cox and Reid (1987), but with random effects. We refer to the approximated marginal posterior obtained in this fashion as the APL for adjusted profile likelihood.

The standard estimates used from equation 3.1, or the LAMP, are based on a Laplace approximation applied once at $\hat{u}_4$. In this case the approximate marginal likelihood function is defined as

$$f(u_4^i \mid y) \approx \exp\{h(\hat{u}_{1-3}, u_4^i)\} \left| \frac{1}{2\pi} - h''(\hat{u}_{1-3}, \hat{u}_4) \right|^{1/2} \tag{3.6}$$

where $\hat{u}_[1-4]$ are the MLEs. In the LAMP equation 3.6, the marginal likelihood is assumed to be proportional to a normal density function with mean $\hat{u}_4$ and variance equal to $|-h''(\hat{u}_{1-3}, \hat{u}_4)|$. However, in our approximation procedure using 3.5, this is not true. Here we assume that for any fixed value of the random effect of interest, u_4^i, the likelihood function is well approximated by a scaled multivariate normal density with mean $\hat{u}_{1-3}^i$ and variance $-h''(\hat{u}_{1-3}^i, u_4^i)$, but the form of the marginal posterior of u_4 is not restricted to be a scaled normal.

In Figure 3.3 we see an example approximated marginal posterior for a particular gene. In this case symmetric bounds would have included zero (and the gene called "neutral"), but bounds found using our method result in a "negative" classification of $\beta^{ND} + \beta^{NDG}$.

The results included in the following section used 500 values of u_4^i ranging from $\hat{u}_4 \pm 5$ standard errors (standard errors taken to be those calculated in section 2.2). By "profiling" the marginal posterior in this way we hope to improve our estimates of the confidence bounds over the standard approximation.

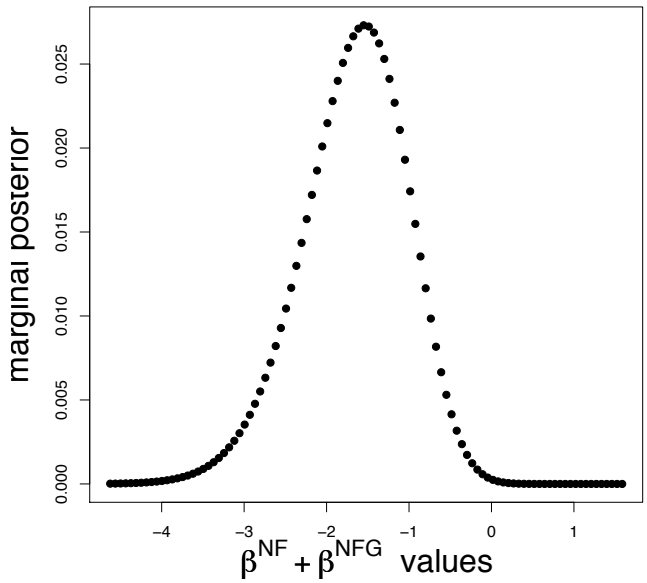


Figure 3.3: **Example results for marginal posterior approximation via adjusted profile likelihood.** The posterior is approximated at many values of the selection effect.

3.2.2 Assessment

To assess our approximation we would like to compare our estimated marginal posterior to the "true" marginal posterior. However, even for simulated data this is unknown. Our fully Bayesian implementation, however, does yield marginal posterior distributions. In the procedure outlined above, we assume that the global parameters β, β^N, β^D, β^{ND}, and Σ are known. Thus, fixing our global parameters to be those estimated by the Bayesian model, we can compare our APL approximated marginal posterior to that of the "true" Bayesian model.

The Bayesian confidence bounds are constructed such that 2.5% of the posterior distribution lies above and below the bounds, the same way our APL

bounds were constructed using numerical quadrature. In Figure 3.4 A we plotted the difference between the APL bounds and the "true" Bayesian bounds, and see that the APL bounds appear to be fairly unbiased estimates. In fact, in this data set (human-chimp comparison data) classification of the selection effect based on the APL had over 99% concordance with the classification based on the MCMC posterior samples.

We then calculated the APL bounds using the parameters estimated in the empirical Bayes setting. In Figure 3.4 B we plotted the difference between the APL bounds and the LAMP bounds. If we assume the APL approximation is again unbiased, then it appears the LAMP bounds are biased. The concordance between the selection effect classification based on the APL approximation and the classification based on the LAMP was only 92%. While this is still fairly high, the APL approximation had a nearly 50% increase in the number of genes classified as under negative selection.

Our APL approximation integrates over the β^G, β^{GN}, and β^{GD}, but treats the random effects covariance, Σ, and the fixed effects, β, β^N, β^D, β^{ND}, as known. The fixed effects tend to be very stable and agree well between the Bayesian and EB methods (all estimates are within 0.02). However, the LAMP bounds yield only a 26.7% concordance with the Bayesian results. The APL bounds increase the agreement to 33.2%. The remaining (large) discrepancy between the two methods is due to the differences in the Bayesian and empirical Bayes estimates of Σ. In Figure 3.5 we see that the two estimates differ non-trivially, with the SnIPRE method yielding larger variance estimates. Not surprisingly this results in larger confidence intervals in the SnIPRE model and those genes identified as under selection are a subset of those identified in the B SnIPRE implementation.

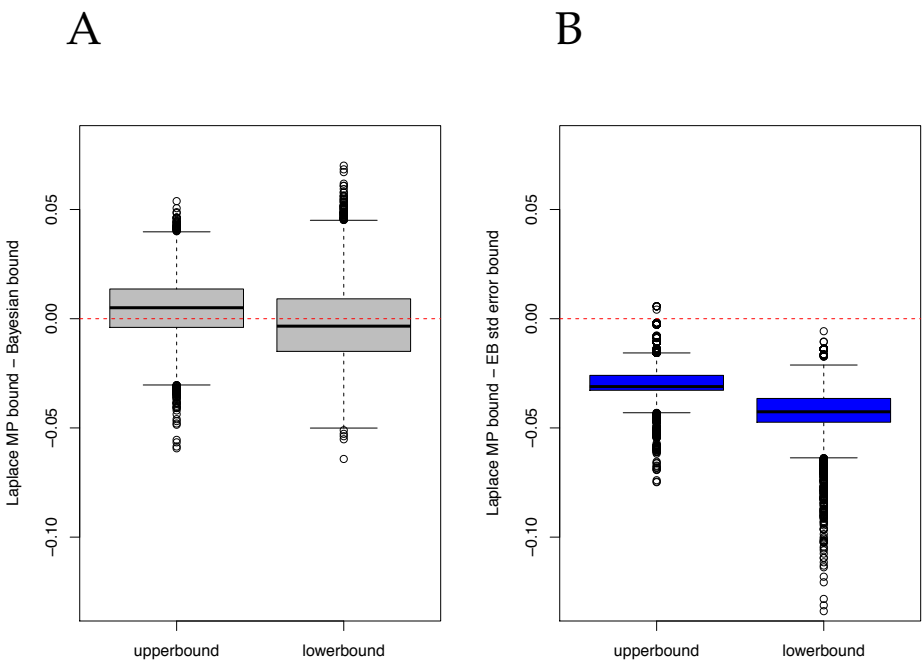


Figure 3.4: **Bias.** Little to no bias in the Bayesian example on left. Assuming then, that the APL method is a good approximation, we see that the standard errors are quite biased. As in the figure 2.3, the APL bound accounts for the left skewed marginal posterior.

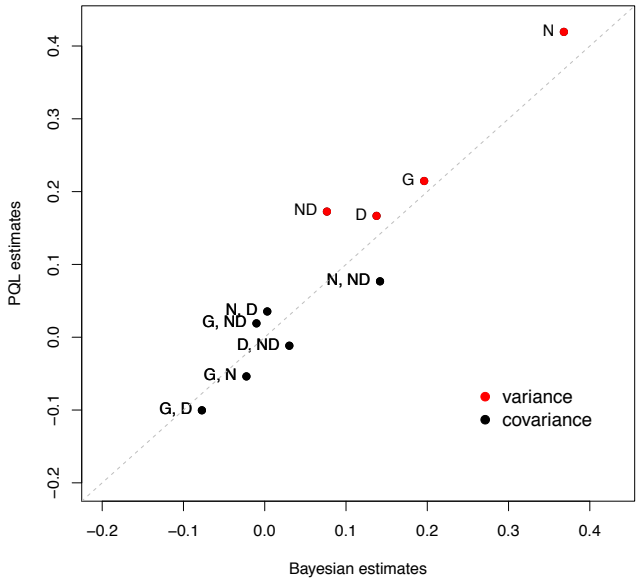


Figure 3.5: **Empirical Bayes versus Bayesian estimates of the variance components (human data).**

3.3 Application

We applied the SnIPRE methods to *Drosophila simulans* data with a *Drosophila melanogaster* out-group. This data was originally presented by Begun et al Begun et al. (2007). Our results are consistent with others' findings of abundant positive selection among *Drosophila* Bierne and Eyre-Walker (2004) Welch (2006) Smith and Eyre-Walker (2002). We also find evidence of a significant amount of mutational constraint, see Figure 3.6. These results are consistent with the large effective population size of Drosophila and the strong efficacy of selection.

In contrast, when we applied SnIPRE to human data, we found few genes with evidence of strong positive selection and an overwhelming signal of negative selection, see Figure 3.7. This is consistent with our previous interpretation of the results in Bustamante et al. (2005) and Boyko et al. (2008), where we argued weak negative selection is the predominant mode of selection operating across the majority of human evolutionary history. Again, this is consistent with the small long term N_e of our species. An implication of this result is that many genes likely harbor mutations of small negative effect that can reach appreciable frequencies.

While there is very little evidence of positive selection based on the selection effect alone in the in the human-chimp comparison, in Figure 3.8 we see that the distribution of constraint estimates, $\hat{f}$, has a strong positive skew. As illustrated in the simulations in section 2.1, this may be evidence of rampant positive selection. The empirical Bayes method has 107 genes with $\hat{f} > 1$ and the Bayesian method has 90 genes with $\hat{f} > 1$.

SnIPRE and B SnIPRE agree quite well on the *Drosophila* data, with 98.5%

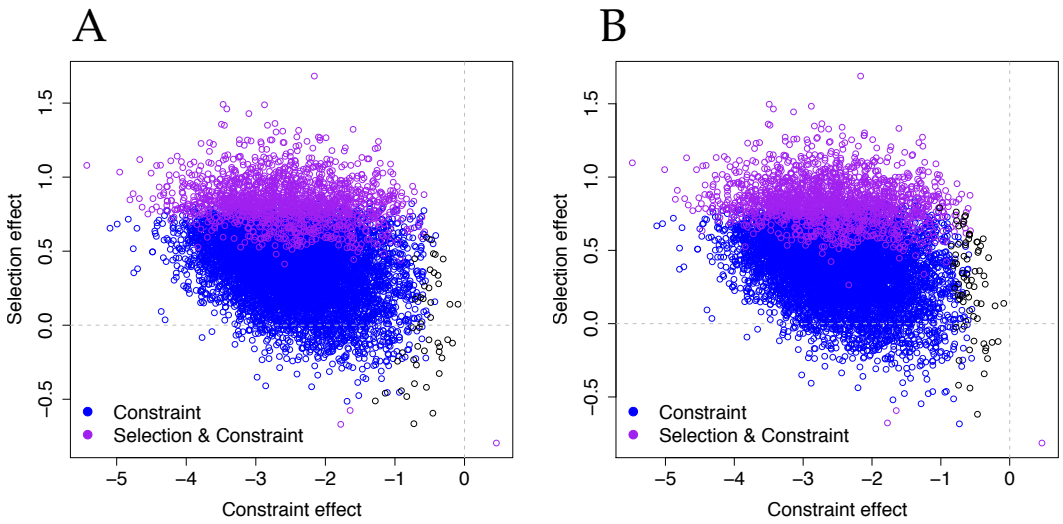


Figure 3.6: ***D. simulans* estimated selection effects and non-synonymous effects for 8,887 genes.** Plots A and B show the estimated selection effects using SnIPRE and B SnIPRE respectively.

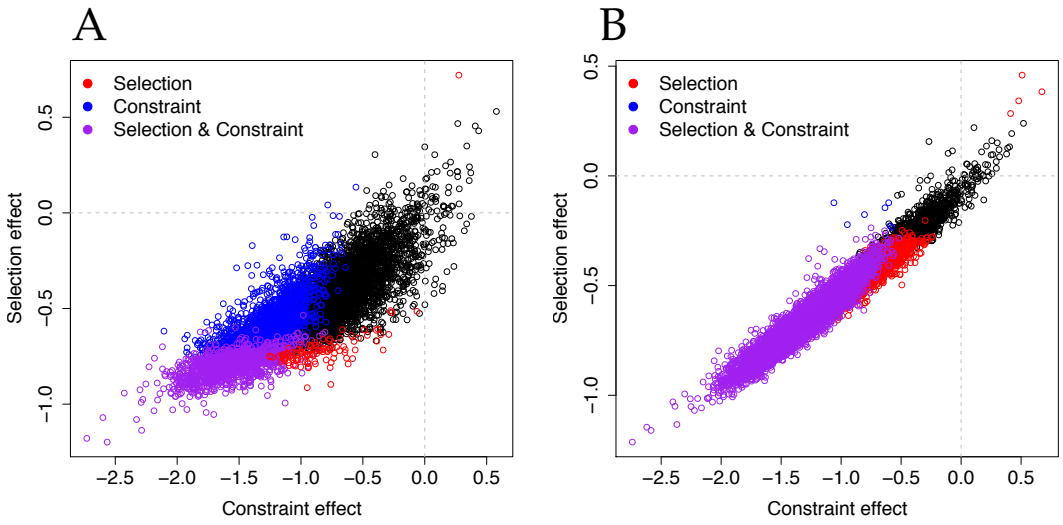


Figure 3.7: **Human estimated selection effects and non-synonymous effects for 11,624 genes.** Plots A and B show the estimated selection effects using SnIPRE and B SnIPRE respectively.

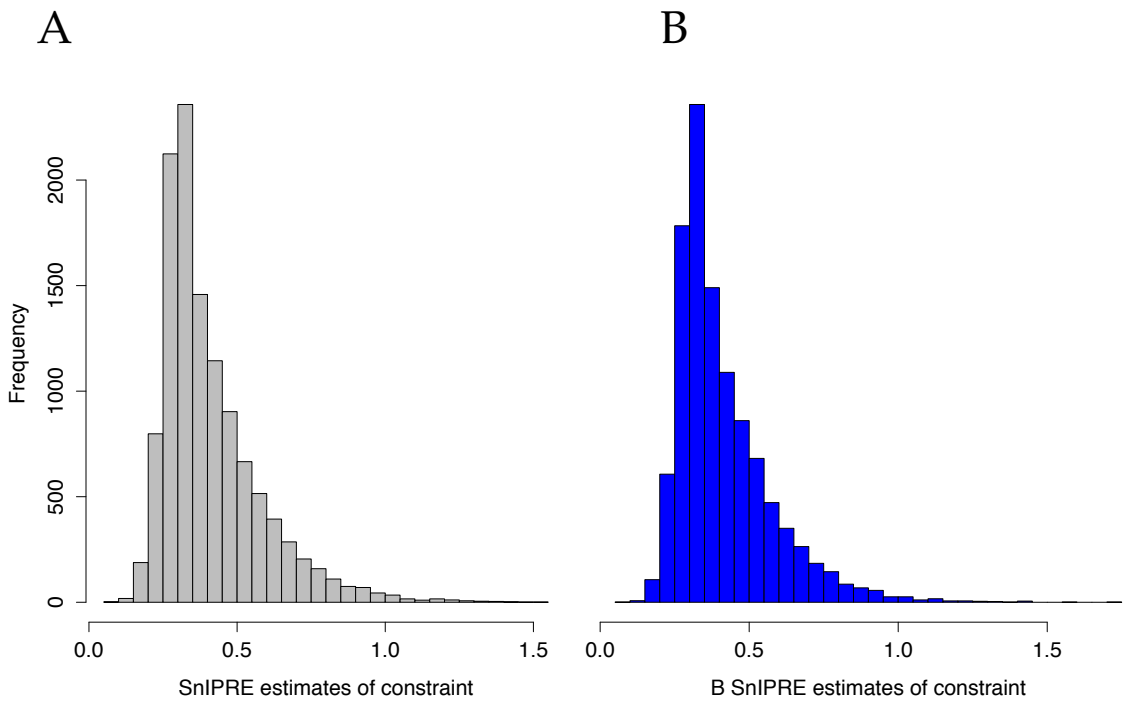


Figure 3.8: **Distribution of estimated constraint in 11,624 human genes.** Plots A and B show the estimated distribution of constraint using SnIPRE and B SnIPRE respectively.

agreement in classification. However, the agreement is only 26.7% on the human data (note that this large discrepancy is mainly in the classification as the selection effect estimates agree fairly well between the two implementations: mean of the estimate differences = 0.017, variance of the estimate differences = 0.003). The higher concordance in the drosophila results is not surprising since the *Drosophila* mutation counts were generally much higher (due to a much higher mutation rate), resulting in more information and more stable estimates. The lower mutation rate in humans results in lower mutation counts. Mutational constraint and negative selection compound this effect. Using the more precise confidence bounds based on the marginal posterior, we increase the agreement between SnIPRE and B SnIPRE from 26.7% to 33.2%. The re-

maining discrepancy is accounted for by the differing estimates of covariance, as mentioned in the previous section.

CHAPTER 4

DISCUSSION

Identifying locations in the genome that have evolved under significant natural selection is a question that has driven the development of a slew of statistical tests. The MK test is popular due to its robustness to demography, limited assumptions, and congruity with population genetic theory. However, with the surfeit of genetic comparison data now available genome-wide, a test based on such a limited amount of the information available is wildly underpowered. In Chapter 1 we introduced SnIPRE, a modeling framework that allows us to take advantage of the abundance of information as well as conserve the desirable aspects of the MK test.

The SnIPRE framework estimates distributions of the number of mutations in various classes, but makes no assumptions about the underlying population genetics model. However, in section 1.2.3 we showed that the estimates from the SnIPRE model can be viewed as a re-parameterization of the Poisson random field framework when the additional assumptions are met. The simulations in section 1.3 illustrated the effectiveness of the SnIPRE methodology, the large increase in power over the MK statistic, and the competitiveness of the parameter estimates with the more parametric model MKprf.

In section 2.1 we explored the robustness of the SnIPRE estimates of selection to the assumption of independent sites. Our results imply that when this assumption is not met we are likely to underestimate the average selection coefficients. Another consequence of a low recombination rate is an overestimation of the constraint effect due to an increase in segregating sites caused by positive selection. For this reason, further insights into the effects of natural selection

may be gained by examining the joint distributions of the selection and constraint effects. Generally, if the constraint effect is believed to be at some value lower than estimated (which is likely true in the case where $\hat{f} > 1$), then the corresponding estimate of selection will increase.

SnIPRE can be implemented in both the empirical Bayes and Bayesian platforms. Generally, the results presented here show the two methods in near perfect agreement. However, the most challenging aspect of fitting the model is estimating the covariance matrix of the random effects, Σ. In the simulations and as well as the *Drosophila* results discussed in section 2.3 the estimated Σ were quite similar between the two methods. Because the confidence intervals for the random effects is a function of Σ the classification of genes under natural selection was largely consistent in these cases. However, the estimated Σ for the human-chimp comparison data had (seemingly small) differences that were large enough to result in differential classification of nearly 75% of the genes, despite strong agreement in the actual estimates of the selection effect. In section 2.2 we describe a method for more precisely estimating the confidence bounds by using the APL approximation to the marginal posterior of the random effects in place of the standard LAMP approximation. Using this more precise approximation improved the discordance rate from 75% to 67% discordance. While initially concerning, the ramifications of relying on one implementation over the other may be minimal depending on purpose of the analysis, since the distribution of the estimated selection effects and the estimates themselves are largely concordant.

Putting the issue of covariance estimation in the case of the human-chimp comparison aside for the moment, generally speaking the Bayesian method-

ology has the advantage in building confidence intervals, examining the joint distributions of random effects, and the potential to incorporate prior knowledge in to the priors. The empirical Bayes implementation, however, is quite a bit faster and a little simpler to implement. The advantage of speed, however, disappears when APL estimates are used to construct the confidence intervals (the two methods are roughly equivalent time-wise in this case).

If classification of genes is the primary concern, future work may include a model similar to that discussed in the following chapter. This type of model would incorporate a latent variable indicating "positive", "negative", or "null" selection effect. An additional latent variable for the constraint effect (likely simply "constrained" or "not constrained") may also be of interest. This type of classification is similar to a strategy employed by a version of MKprf which groups genes into a pre-determined number of classes based on the evidence for a particular gene to have certain level of selection, but the SnIPRE implementation would avoid assuming a particular underlying population genetics model.

Acknowledgments

Adam Boyko, Ryan Hernandez, and the Bustamante Lab, especially Jeremiah Degenhardt and Kirk Lohmueller.

CHAPTER 5

PROWLRE: PROTEOMICS WITH LATENT VARIABLES AND RANDOM EFFECTS

In biological analyses one commonly occurring general design consists of :

- Two treatments: "control" and "case"
- Hundreds, or possibly thousands of observational units
- Replicates

Many microarray, fMRI, RRBS methylation, and label free shotgun proteomics data sets have this design, and the analyses aim to identify which observational units have differentiated responses between the two groups. In label free shotgun proteomics analyses, for example, the goal is to identify those proteins with differentiated abundance in the "case" and "control" treatments. These two treatments may refer, for example, to two different conditions, or cellular states. The units are the proteins, and the replicates are spectral counts from different samples.[1] The analysis can be viewed as a two-group classification problem, in which the proteins are assigned to the "null" group if there is no difference in the case an control abundance, and the "non-null" group if there is differentiated abundance. This two-group classification approach is widely accepted in the microarray literature (Bar et al., 2010; Efron, 2008), and its application in shotgun proteomics data is similarly pertinent.

In the paper Booth, Eilertson, Olinares, and Yu (2011, hereafter MCP 2011) we introduced a fully Bayesian implementation for classifying proteins as "null"

[1]Spectral counts have become an accepted stand in for protein abundance, as they have been shown to correlate well with the abundance of the corresponding protein (Liu et al., 2004; Cooper et al., 2010; Old et al., 2005; Zhang et al., 2009).

or "non-null" in label free shotgun proteomics analyses. We also reviewed commonly used one at a time methods (score, likelihood-ratio, and Wald tests), as well as a new method presented in Choi et al. (2008) named "Qspec", and compare the performance of these methods to our Bayesian model analysis. We showed that the Qspec analysis can be viewed as version of the LRT test arrived at using Bayesian techniques, and also that our model offers a significant increase in power over these traditional tests.

The Bayesian analysis we presented in MCP 2011 relies on MCMC techniques (implemented in OpenBUGS). In this chapter we introduce an empirical Bayes implementation of the model which can be fit using the EM algorithm. Our strategy is similar to that employed in the LEMMA algorithm proposed by Bar et al. (2010) for the microarray data analysis case. However, we face the additional difficulty of a non-gaussian response which results in an analytically intractable complete data likelihood. We address the resulting challenges through use of the Laplace approximation and the iteratively re-weighted least squares (IRLS) approach. The model is described in section 5.1, and the algorithm is detailed in 5.2. In section 5.3 we compare the results of our Expectation Maximization (EM) algorithm estimation to those of the fully Bayesian model fit using OpenBUGS.

There are two main advantages to our Bayesian and empirical Bayes analyses. The first is the ability to share information across proteins. As B. Efron points out, the data set's "massively parallel structure, with thousands of *similar* situations each providing information allows an appropriate prior distribution to be estimated from the data" (Efron, 2008). Our Bayesian model methods take advantage of the plethora of information by using it to estimate the global

model parameters, θ. The protein specific variables b_k are estimated based on combination of the global information and subject specific observations. Combining information in this fashion provided a substantial increase in power for our fully Bayesian analysis employed in our MCP 2011 paper, and in section 5.3 we illustrate that the empirical Bayes analysis offers the same advantage.

The second advantage of using our Bayesian classification method is that we are quantifying evidence that more directly addresses the question of interest: Which proteins have differential abundance? By including a latent variable indicating the null or nonnull status of the protein, we are in fact estimating the quantity we are interested in, namely the P(protein k is nonnull). In MCP 2011 we do this by looking at the posterior distributions of the latent variables, and here we accomplish this by using Bayes rule and the likelihood of the data under the null and nonnull settings.

As we noted in MCP 2011 one drawback to using the Bayesian model for inference is the increase in computation time required for fitting the model, as well as the additional necessary step of implementing convergence diagnostics. The EM algorithm as we have outlined it below is potentially a faster and more stable method for fitting the latent variable model.

5.1 Model

The data consists of spectral counts, y_{jk}, where $j = 1, \ldots, n$ indicates the replicate and $k = 1, \ldots, p$ indicates the protein. In our model the spectral counts, y_{jk} are assumed to follow a Poisson distribution with mean μ_{jk}, given the treatment T_j, the latent variable I_k, and the protein specific random effects b_k. Our model also

accounts for sample (j) effects and protein length in the offset O_{jk}.

$$y_{jk} \mid T_j, I_k, b_k \sim Poisson(\mu_{jk})$$

$$\log(\mu_{jk}) = \beta_0 + \beta_1 T_j + \beta_2 T_j I_k + b_{0k} + b_{1k} T_j I_k + O_{ijk} \tag{5.1}$$

$$b_{0k} \sim N(0, \sigma_0^2)$$

$$b_{1k} \sim N(0, \sigma_1^2)$$

$$I_k \sim Bin(p, \pi)$$

Here $T_j = 0$ if the j^{th} observation is a control replicate, and $T_j = 1$ if the observation is a case replicate. According to this model, proteins have a multiplicative treatment effect equal to e^{β_1}, while proteins in the nonnull group ($I_k = 1$) have an additional multiplicative treatment effect of e^{β_2}. The random effect b_{ok} accounts for over-dispersion due to protein specific effects, and b_{1k} accounts for additional over-dispersion due to treatment effects on individual proteins.

Using this model we can make inference about the probability a particular protein has differential abundance among treatments $P(I_k = 1)$ while accounting for variability among proteins, and an overall treatment effect. In the fully Bayesian setting this is done by examining the posterior distribution of the latent variables I_k. Here, we calculate the posterior expectations using Bayes rule:

$$P(I_k = 1 \mid y_k) = \frac{\pi f(y_k \mid I_k = 1)}{\pi f(y_k \mid I_k = 1) + (1 - \pi) f(y_k \mid I_k = 0)}$$

In this way we are able to do simultaneous testing of several thousand proteins through the calculation of posterior probabilities of their null and non-null status.

5.2 Method

Fitting model 5.1 can be viewed as a missing data problem with the complete data represented by (y, I), where $y = (y_1', \ldots, y_p')$ and the missing data are the latent variables $I = (I_1, \ldots, I_p)$ indicating null or non-null group status. The maximum likelihood estimates of the model parameters, $\theta = (\beta_0, \beta_1, \beta_2, \sigma_0^2, \sigma_1^2)$, and the latent variables, I_k, can be found using the EM algorithm (Dempster et al., 1977). Our method is outlined below.

E-step:

The expectation step is evaluated using our current estimates of the latent variables: $P(I_k = 1|y_k) = p_{1k}$ and $P(I_k = 0|y_k) = 1 - p_{1k}$.

$$E_{I|y,\theta^t}\left[L_c(\theta; y, I)\right] = Q(\theta, \theta^t)$$

Where L_c is the likelihood for model 5.1:

$$L_c = \prod_k \left\{(1-\pi)\int P_0(y_{jk}; \mu_{jk}^0)\phi(b_k; D)db_k\right\}^{(1-I_k)} \times \prod_k \left\{\pi \int P_1(y_{jk}; \mu_{jk}^1)\phi(b_k; D)db_k\right\}^{I_k} \tag{5.2}$$

Where $P_r(y_{jk}; \mu_{jk}^r) = P(y_{jk}; \mu_{jk} \mid I_k = r)$ is the Poisson mass function for y_{jk} with mean μ_{jk}^r, $r = 0, 1$; $\phi(\cdot; D)$ is the multivariate normal mass function with mean zero and covariance $D = \begin{pmatrix} \sigma_0^2 & 0 \\ 0 & \sigma_1^2 \end{pmatrix}$. Our Q-function (using the log-likelihood), can thus be written

$$\begin{aligned} Q = E(l_c(\theta) \mid y) \quad &= \quad \sum_r \sum_k p_{rk} \log \int P_r(y_k; \mu_k)\phi(b_k; D)db_k \\ &\quad + \sum_k \left\{(1-p_{1k})\log(1-\pi) + p_{1k}\log\pi\right\}. \end{aligned} \tag{5.3}$$

From the form of the likelihood in 5.2, we see we are fitting a mixture of two models - one model for the null case ($r = 0$), equation 5.4, and one model for the

nonnull case ($r = 1$), equation 5.5.

$$\log(\mu^0_{jk}) = \beta_0 + \beta_1 T_j + \beta_2 T_j 0 + b_{0k} + b_{1k} T_j 0 + O_{ijk} \tag{5.4}$$

$$\log(\mu^1_{jk}) = \beta_0 + \beta_1 T_j + \beta_2 T_j 1 + b_{0k} + b_{1k} T_j 1 + O_{ijk} \tag{5.5}$$

Let us define the marginal likelihood of model r for protein k as

$$\begin{aligned} f_r(y_k) &= \int P_r(y_k; \mu_k)\phi(b_k; D)db_k \\ &= \int \left\{ \prod_j \frac{\mu_{jk}^{y_{jk}}}{y_{jk}!} e^{-\mu_{jk}} \right\} \mid 2\pi D \mid^{-1/2} e^{-\frac{1}{2} b_k' D^{-1} b_k} db_k \\ &= \int e^{\sum_j \left(y_{jk} \log \mu^r_{jk} - \mu^r_{jk} \right) - \frac{1}{2} b_k' D^{-1} b_k} db_k \mid 2\pi D \mid^{-1/2} \prod_j \frac{1}{y_{jk}!} \\ &= \int e^{h_r(b_k)} db_k \mid 2\pi D \mid^{-1/2} \prod_j \frac{1}{y_{jk}!} \end{aligned} \tag{5.6}$$

Where

$$h_r(b_k) = \sum_j \left(y_{jk} \log \mu^r_{jk} - \mu^r_{jk} \right) - \frac{1}{2} b_k' D^{-1} b_k.$$

The integral in 5.6 is analytically intractable, but may be approximated using the Laplace approximation:

$$\tilde{f}_r(y_k) \approx e^{h_r(\tilde{b}_k)} \mid 2\pi \tilde{H}_r \mid^{1/2} \mid 2\pi D \mid^{-1/2} \prod_j \frac{1}{y_{jk}!} \tag{5.7}$$

Where $\tilde{b}_k$ are the values for the variables of integration that maximize h_r, and are easily calculated using the Newton Raphson method, and $-\tilde{H}^{-1}$ represents the matrix of partial second derivatives, or Hessian matrix, evaluated at $\tilde{b}_k$. The Hessian is a function of μ^r_{jk} and so varies by level of $(T_j I_k)$.

$$\begin{aligned} H_r^{-1} &= Z^{r\prime} W^r Z^r + D^{-1} \\ &= \sum_j \mu^r_{jk} z^r_{jk} z^{r\prime}{}_{jk} + D^{-1} \end{aligned}$$

$$z'_{jk} = \begin{cases} [1 \;\; 1] & \text{when } r = T_j = 1 \\ [1 \;\; 0] & \text{when } o.w. \end{cases}$$

Thus

$$
\begin{aligned}
H_0^{-1} &= \sum_j \mu_{jk}^0 \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + D^{-1} \\
H_1^{-1} &= \sum_j \mu_{jk}^1 \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}^{(1-T_j)} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}^{(T_j)} + D^{-1}
\end{aligned}
$$

Using the Laplace approximated marginal likelihoods, we can estimate the posterior probability of protein k being non-null by applying Bayes rule. This leads to the update of our latent variable parameters for the $t+1$ iteration:

$$
\begin{aligned}
p_{1k}^{t+1} &= \frac{\pi^t f_1(y_k \mid \theta^t)}{\pi^t f_1(y_k \mid \theta^t) + (1-\pi^t) f_0(y_k \mid \theta^t)} \\
&\approx \frac{\pi^t \tilde{f}_1(y_k \mid \theta^t)}{\pi^t \tilde{f}_1(y_k \mid \theta^t) + (1-\pi^t) \tilde{f}_0(y_k \mid \theta^t)}
\end{aligned} \tag{5.8}
$$

M-step:

$$
\theta^{t+1} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta, \theta^t)
$$

After conditioning on the latent variables we estimate θ such that the likelihood, Q, for the mixture of the generalized linear mixed effects models is maximized. We accomplish this by fitting these two models (5.4, 5.5) simultaneously while using the probabilities from the E-step, 5.8, to weight the observations according to their probabilities of belonging to the model group.

The Q-function from 5.3 has two parts which can be maximized separately

$$
\sum_r \sum_k p_{rk} \log \int P_r(y_k; \mu_k) \phi(b_k; D) db_k \tag{5.9}
$$

and

$$
\sum_k (1 - p_{1k}) \log(1-\pi) + p_{1k} \log \pi. \tag{5.10}
$$

The maximization of the second part 5.10 leads to the straight-forward update of π

$$\pi^{t+1} = \frac{1}{p}\sum_k p_{1k}^{t+1}.$$

The first part, 5.9 can be written more explicitely as

$$\sum_r \sum_k p_{rk} \log \int e^{\sum_j \left(y_{jk}\log\mu_{jk}^r - \mu_{jk}^r\right) - \frac{1}{2}b_k' D^{-1} b_k} db_k \mid 2\pi D \mid^{-1/2} \prod_j \frac{1}{y_{jk}!}.$$

Replacing the integrals in the first part with the Laplace approximation from 5.7 we get

$$\begin{aligned}
(5.9) \quad &\approx \sum_r \sum_k p_{rk} \log \left[e^{h_r(\tilde{b}_k)} \mid 2\pi\tilde{H}_r \mid^{1/2} \mid 2\pi D \mid^{-1/2} \prod_j \frac{1}{y_{jk}!} \right] \\
&\propto \sum_r \sum_k p_{rk} \left[h_r(\tilde{b}_k) + \log \mid 2\pi\tilde{H}_r \mid^{1/2} + \log \mid 2\pi D \mid^{-1/2} \right] \\
&= \sum_r \sum_k p_{rk} \left[\sum_j \left(y_{jk}\log\tilde{\mu}_{jk}^r - \tilde{\mu}_{jk}^r \right) - \frac{1}{2}\tilde{b}_k' D^{-1}\tilde{b}_k + \log \mid 2\pi\tilde{H}_r \mid^{1/2} + \log \mid 2\pi D \mid^{-1/2} \right]
\end{aligned} \tag{5.11}$$

Ignoring the dependence of $\tilde{H}_r$ on β and fixing the random effects b_k, maximizing

$$\sum_r \sum_k p_{rk} \left[\sum_j \left(y_{jk}\log\mu_{jk}^r - \mu_{jk}^r \right) \right] \tag{5.12}$$

with respect to β is equivalent to fitting a log-linear model with an offset. This can be accomplished by iteratively re-weighted least squares (IRLS) fitting of the linear model

$$z = X\beta + Z\tilde{b} + e \tag{5.13}$$

Here z is the working response

$$z_{jk} = \log\mu_{jk} + \frac{y_{jk} - \mu_{jk}}{\mu_{jk}}$$

and

$$e \sim N(0, W^{-1})$$

$$W = diag(\mu^0 p_{0k},\ \mu^1 p_{1k})$$

Using the penalized quasi- likelihood method of estimation (Schall, 1991; Breslow and Clayton, 1993) the update to β is the solution to Henderson's equations.

$$(X^T W X)\hat{\beta} = X^T W(z - Z\hat{b}) \tag{5.14}$$

$$(Z^T W Z + [D \otimes I_p]^{-1})\hat{b} = Z^T W(z - X\hat{\beta}) \tag{5.15}$$

Iterating between equations 5.14 and 5.15 until convergence. However, we want $b = (b_0^0, b_0^1, b_1^1)$ such that b^r maximizes h_r. Fixing $\hat{b}$ to be $\tilde{b}$, the maximizer of our h function, we get β^{t+1} is the solution to the IRLS equation

$$(X^T W X)\hat{\beta} = X^T W z \tag{5.16}$$

Thus we iterate between 5.16 and maximizing h_0 and h_1. To do this, we made use of the standard algorithm available in R in for fitting generalized linear models, the glm() command. The simultaneous fitting is accomplished by stacking the data and creating the design matrices such that the matrices for the top data set correspond to the null model and the bottom correspond to the nonnull model:

$$[X\ Z] = \begin{bmatrix} X^0_{np\times 3} & Z^0_{np\times 2p} \\ X^1_{np\times 3} & Z^1_{np\times 2p} \end{bmatrix}$$

$$X^0_{jk} = \begin{bmatrix} 1 & T_j & 0 \end{bmatrix}$$

and

$$X^1_{jk} = \begin{bmatrix} 1 & T_j & T_j \end{bmatrix}$$

The random effects design matrix Z^0 should yield

$$[Z\tilde{b}]^0_{jk} = \tilde{b}_{0k} + 0$$

and Z^1 should yield

$$[Z\tilde{b}]^1_{jk} = \tilde{b}_{0k} + \tilde{b}_{1k}$$

The appropriate error variance matrix, W^{-1} is accomplished by weighting the observations from the k^{th} protein in the top portion of the stacked data by p_{0k}, and those in the bottom portion of the stacked data by p_{1k}. So our updates, β^{t+1} are the estimates of the log-linear model

$$\log \mu = X\beta$$

with observations weighted by p_{rk}, and offset $Z\tilde{b}$.

To estimate the variance components, we maximize the approximate log-likelihood from 5.11 using our current estimates of of the fixed and random effects. We accomplished this by using univariate Newton-Raphson updates for roots of the first derivatives of the approximate log-likelihood.

$$\frac{\partial l}{\partial \sigma_0^2} = \sum_r \sum_k p_{rk} \left[\frac{\tilde{b}_{0kr}^2}{\sigma_0^4} - \frac{1}{\sigma_0^2} + tr\left(-\frac{\partial \tilde{H}_r^{-1}}{\partial \sigma_0^2} \tilde{H}_r \right) \right] \tag{5.17}$$

$$\frac{\partial l}{\partial \sigma_1^2} = \sum_r \sum_k p_{rk} \left[\frac{\tilde{b}_{1kr}^2}{\sigma_1^4} - \frac{1}{\sigma_1^2} + tr\left(-\frac{\partial \tilde{H}_r^{-1}}{\partial \sigma_1^2} \tilde{H}_r \right) \right] \tag{5.18}$$

Here the expression $tr\left(-\frac{\partial \tilde{H}_r^{-1}}{\partial \sigma_1^2} \tilde{H}_r\right)$ is the derivative of the $\log |\tilde{H}_r|^{1/2}$ term, and can be found using an identity from Searle et al. (1992), appendix M.7. When $r = 0$, $\tilde{b}_{1kr} \equiv 0$, and the trace term is $\frac{1}{\sigma_1^2}$, making the 5.18 sum over the $r = 0$ terms equal to zero. Thus all the information for for the update of σ_1^2 comes from the non-null case, $r = 1$. When $r = 1$, the $tr\left(-\frac{\partial \tilde{H}_r^{-1}}{\partial \sigma_1^2} \tilde{H}_r\right)$ is a function of both σ_0^2 and σ_1^2. However, we update the estimates of σ_0^2 and σ_1^2 by iterating between one-step

univariate Newton-Raphson updates for finding the roots of 5.17 and 5.18.

$$(\sigma_0^2)^{(t+1)_{i+1}} \mid (\sigma_1^2)^{(t+1)_i} = (\sigma_0^2)^{(t+1)_i} - \frac{\sum_r \sum_k p_{rk} \left[\frac{\tilde{b}_{0kr}^2}{\sigma_0^4} - \frac{1}{\sigma_0^2} + tr\left(-\frac{\partial \tilde{H}_r^{-1}}{\partial \sigma_1^2} \tilde{H}_r \right) \right]}{\sum_r \sum_k p_{rk} \left[\frac{-2\tilde{b}_{0kr}^2}{\sigma_0^6} + \frac{1}{\sigma_0^4} + \frac{\partial}{\partial \sigma_0^2} tr\left(-\frac{\partial \tilde{H}_r^{-1}}{\partial \sigma_0^2} \tilde{H}_r \right) \right]}$$

$$(\sigma_1^2)^{(t+1)_{i+1}} \mid (\sigma_0^2)^{(t+1)_{i+1}} = (\sigma_1^2)^{(t+1)_i} - \frac{\sum_r \sum_k p_{rk} \left[\frac{\tilde{b}_{1kr}^2}{\sigma_1^4} - \frac{1}{\sigma_1^2} + tr\left(-\frac{\partial \tilde{H}_r^{-1}}{\partial \sigma_1^2} \tilde{H}_r \right) \right]}{\sum_r \sum_k p_{rk} \left[\frac{-2\tilde{b}_{1kr}^2}{\sigma_1^6} + \frac{1}{\sigma_1^4} + \frac{\partial}{\partial \sigma_1^2} tr\left(-\frac{\partial \tilde{H}_r^{-1}}{\partial \sigma_1^2} \tilde{H}_r \right) \right]}$$

Thus, the update for $(\sigma_0^2\ \sigma_1^2)^{(t+1)}$ is $(\sigma_0^2\ \sigma_1^2)^{(t+1)_k}$ for some reasonable k (we used $k = 10$ for our analyses in section 5.3).

5.3 Simulation Results

The goal our approach is to fit an empirical Bayes version of the latent variable model. In the Bayesian implementation we used a hierarchical model with vague priors on the hyperparameters, thus the posterior distributions were data driven. This is similar to the empirical Bayes technique where the distribution of the random effects is determined by the data. Thus we should expect the two models to agree quite well. To test the success of our method we compared our results to those from the Bayesian implementation of the model. The first comparison uses a "synthetic" data set using a yeast reference proteome in which 48 proteins were spiked using human proteins with 6.7 and 2.2 fmol/μL UPS1, which yielded a 3 fold difference between groups for these 48 proteins. There were a total of 1386 proteins, with 3 replicates in each treatment group (6 observations per protein). This data set was obtained from the Clinical Proteomic Technology Assessment for Cancer (CPTAC) Study 6 (Paulovich et al., 2010). In Figure 5.1 A the receiver operating curves for the two methods are

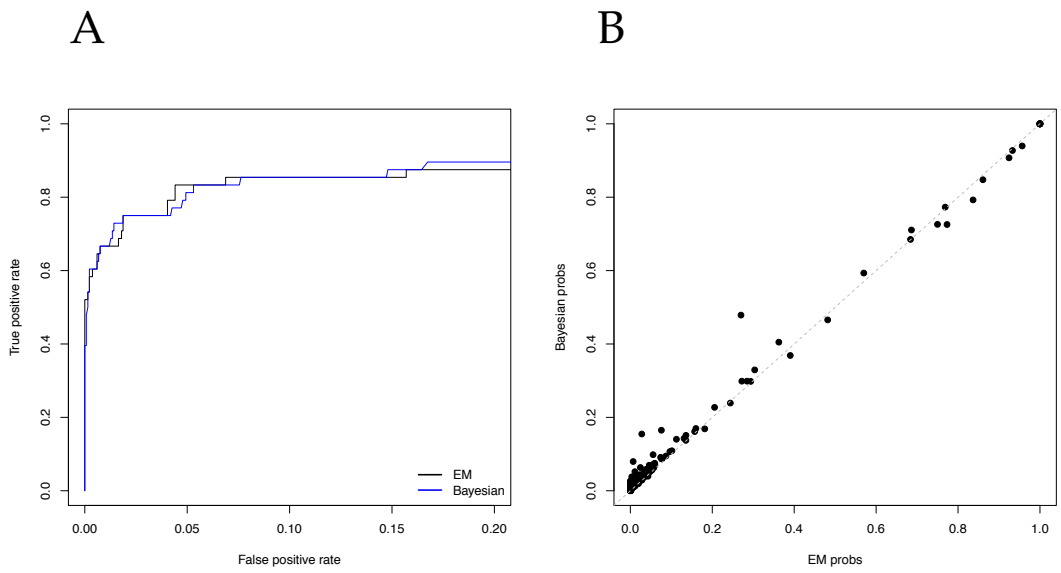


Figure 5.1: **Synthetic data set with Human and Yeast proteins.** Plot A compares the ROC curves between the Bayesian implementation and the EM implementation. Plot B compares the Bayesian and EM estimated p_{1k}.

plotted. In Figure 5.1 B we plotted the the estimated individual protein probabilities p_{1k} from the empirical Bayes implementation against those estimated in the Bayesian. From these plots we see a high concordance in the classification probabilities of null and nonnull status between the two implementations.

We also simulated data using the poisson models 5.4 and 5.5. In this first example, we simulated 1000 proteins with a two-fold increase in the non-null group ($\beta_2 = \log 2$), 4 replicates per treatment (8 observations per protein) and 200 proteins in the nonnull group ($\pi = 0.2$).

We simulated a second example of 1000 proteins with a two-fold decrease in the non-null group ($\beta_2 = \log \frac{1}{2}$), 4 replicates per treatment (8 observations per protein) and 100 proteins in the nonnull group ($\pi = 0.1$), see Figure 5.3. As in the case of the synthetic data, the two simulated data set results show high

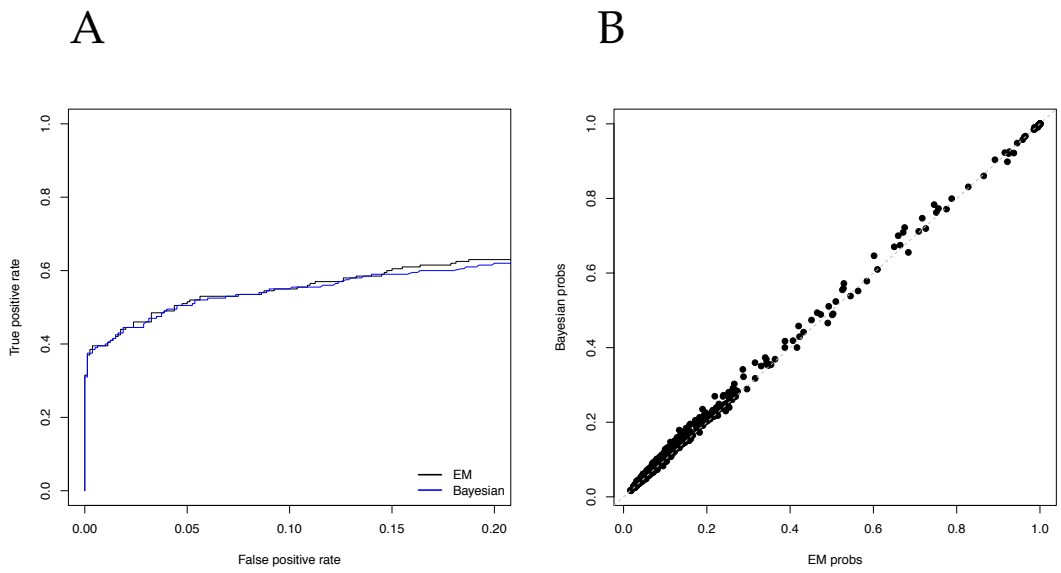


Figure 5.2: **Simulated data set with 2 fold increase for nonnull group.** Plot A compares the ROC curves between the Bayesian implementation and the EM implementation. Plot B compares the Bayesian and EM estimated p_{1k}.

agreement between the empirical Bayes and Bayesian latent variable models.

As mentioned earlier, one of the advantages to an EM algorithm implementation is the potential for an decrease in computation time. For the results presented here we ran the EM algorithm on each data set for 300 iterations which took approximately 20 minutes. This is roughly the time it took to fit the Bayesian model using OpenBUGS. Admittedly, in both situations the computation time may be further improved via more effective programming.

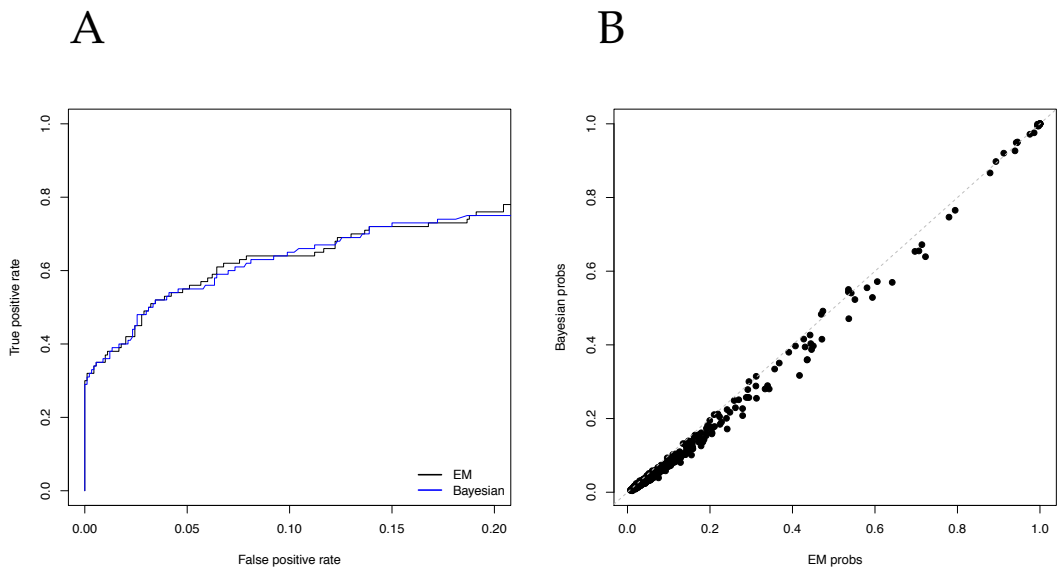


Figure 5.3: **Simulated data set with 2 fold decrease for nonnull group.** Plot A compares the ROC curves between the Bayesian implementation and the EM implementation. Plot B compares the Bayesian and EM estimated p_{1k}.

5.4 Discussion

The model presented here offers a large increase in power over traditional methods for testing for the presence of treatment effects among many observational units, a commonly conducted analysis in microarray, fMRI, methylation, and proteomics data. This increase in power is due to the capitalization on shared information across observational units. As shown in section 5.3, the empirical Bayes implementation presented here offers the same improvement as the fully Bayesian implementation presented in MCP 2011. Another advantage to our modeling framework is the ability to make direct inference on the parameter of interest: the probability a particular unit's response is differentiated between treatment groups. Thus, eliminating the need for the arbitrary "significance cut-offs" of other methods.

The latent variable model can be fit in the EM framework, as presented in section 5.2. The updates to p_{1k}, π, β, and D can be calculated using the Laplace approximation to the likelihood function. We have also explored using a second order Laplace approximation, but found no noticeable improvement. The update to β is achieved through iteratively solving 5.16 and maximizing h_r (for $r = 1, 0$). However, we have found a one-step update (simply the solution to 5.16 using current values of $\tilde{b}_k$) to be a sufficient approximation. The results presented in section 5.3 are achieved in this way.

While the results presented here focus primarily on the classification aspect, the empirical Bayes and Bayesian methods also agree in the estimation of the other model parameters. Convergence plots comparing the EM estimates to OpenBUGS estimates can be found in the appendix.

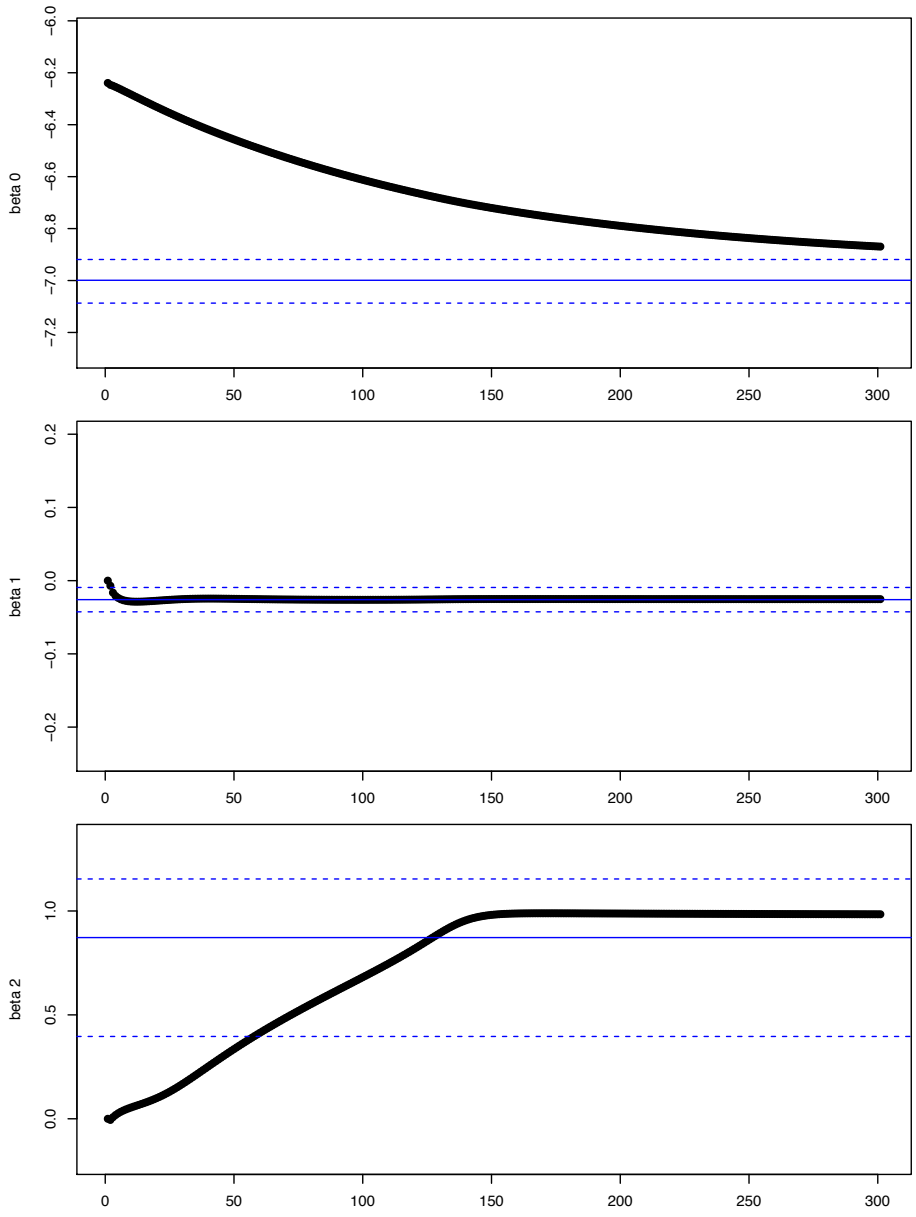
Just as the model and algorithm described here are an extension of that originally presented by Bar et al. (2010), similar adaptations can be implemented for use with other members of the generalized linear model family. Likewise, the LEMMA method latent variable model is set up to handle a three group scenario (I_k = -1, 0, 1), and this too should be fairly straightforward to incorporate into the fitting procedure. In the proteomics analyses, this would extend inference to include statements about the probability a particular protein has the same, increased, or decreased abundance in the treatment samples. Currently, our research is focusing on these extensions.

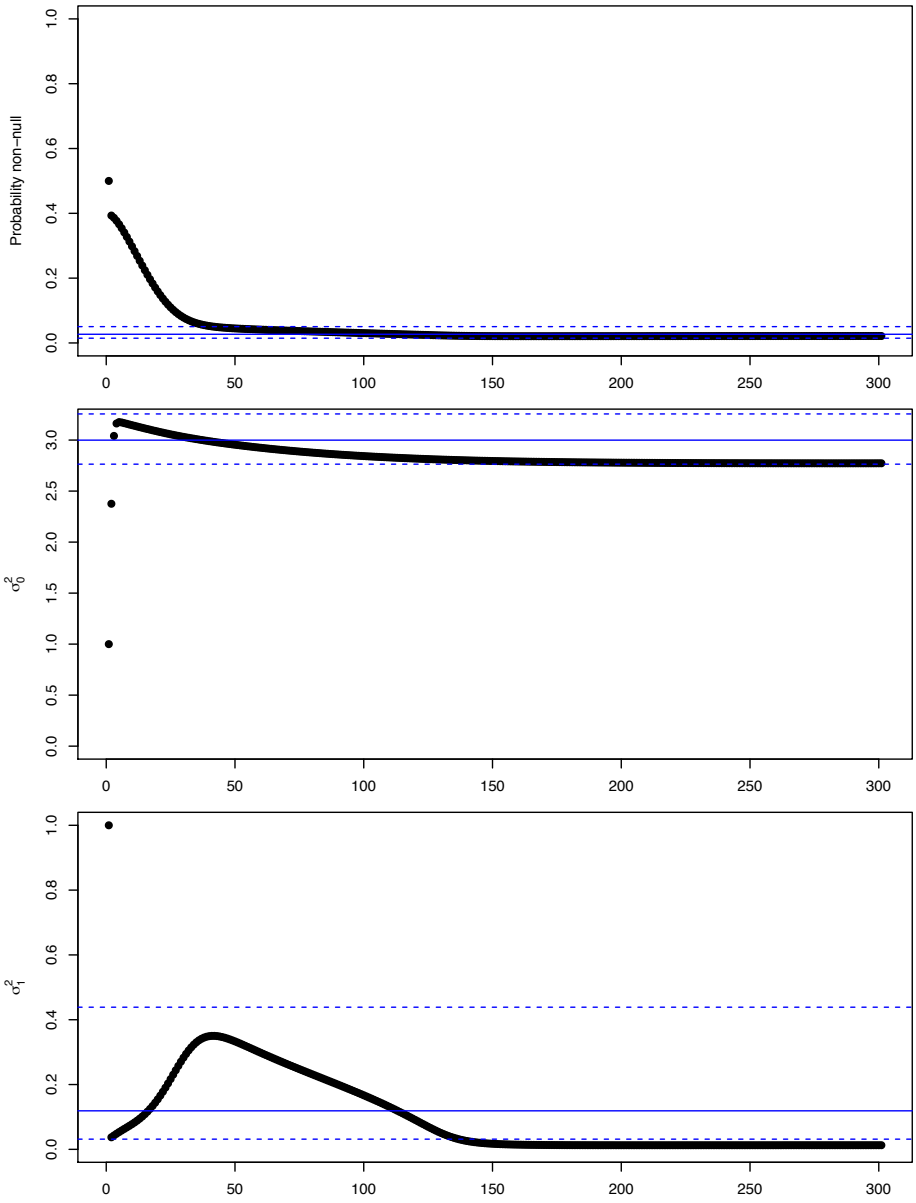
APPENDIX A

PROWLRE CONVERGENCE PLOTS

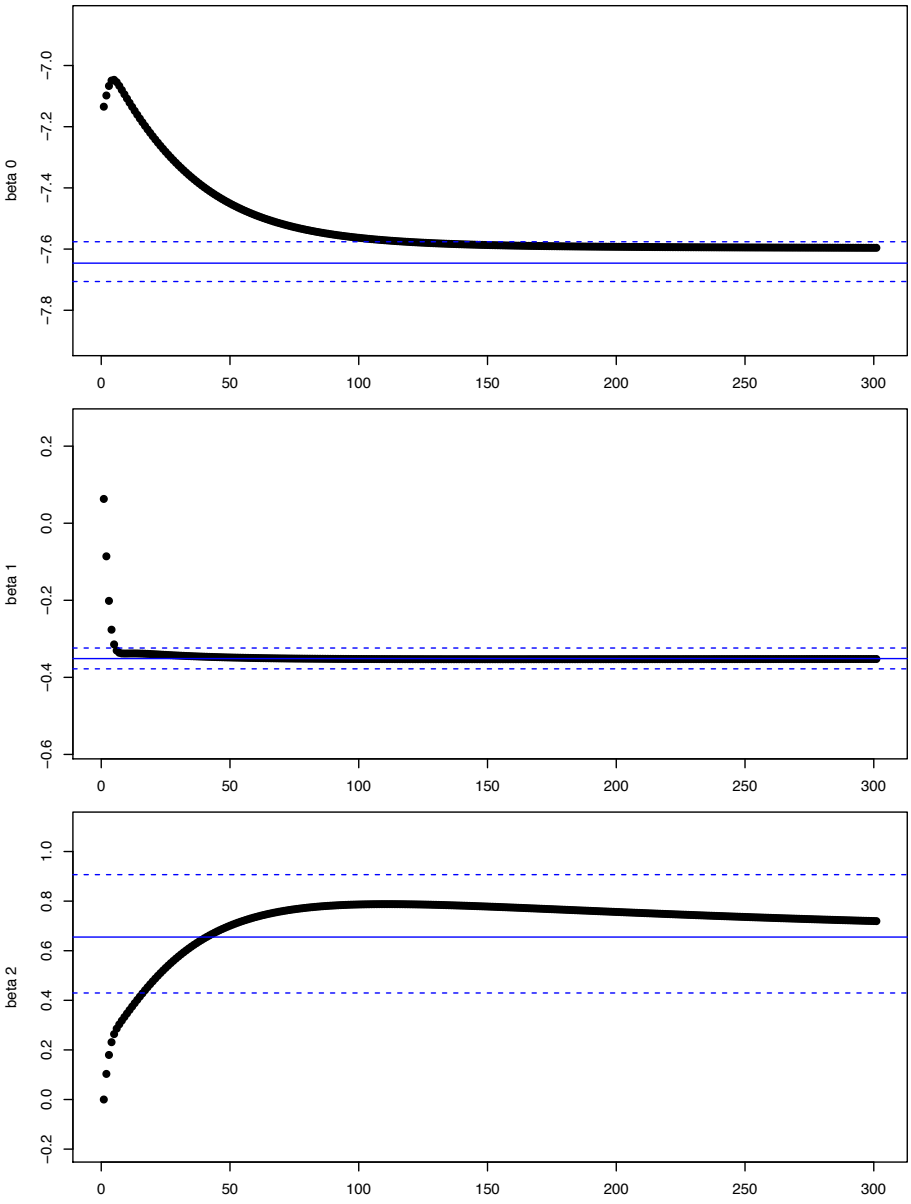
Solid fine line represents the Bayesian estimates, and the dashed fine lines represent the CI bounds for the estimates. The solid circles respresent the empirical Bayes estimates at the iteration indicated by the x-axis.

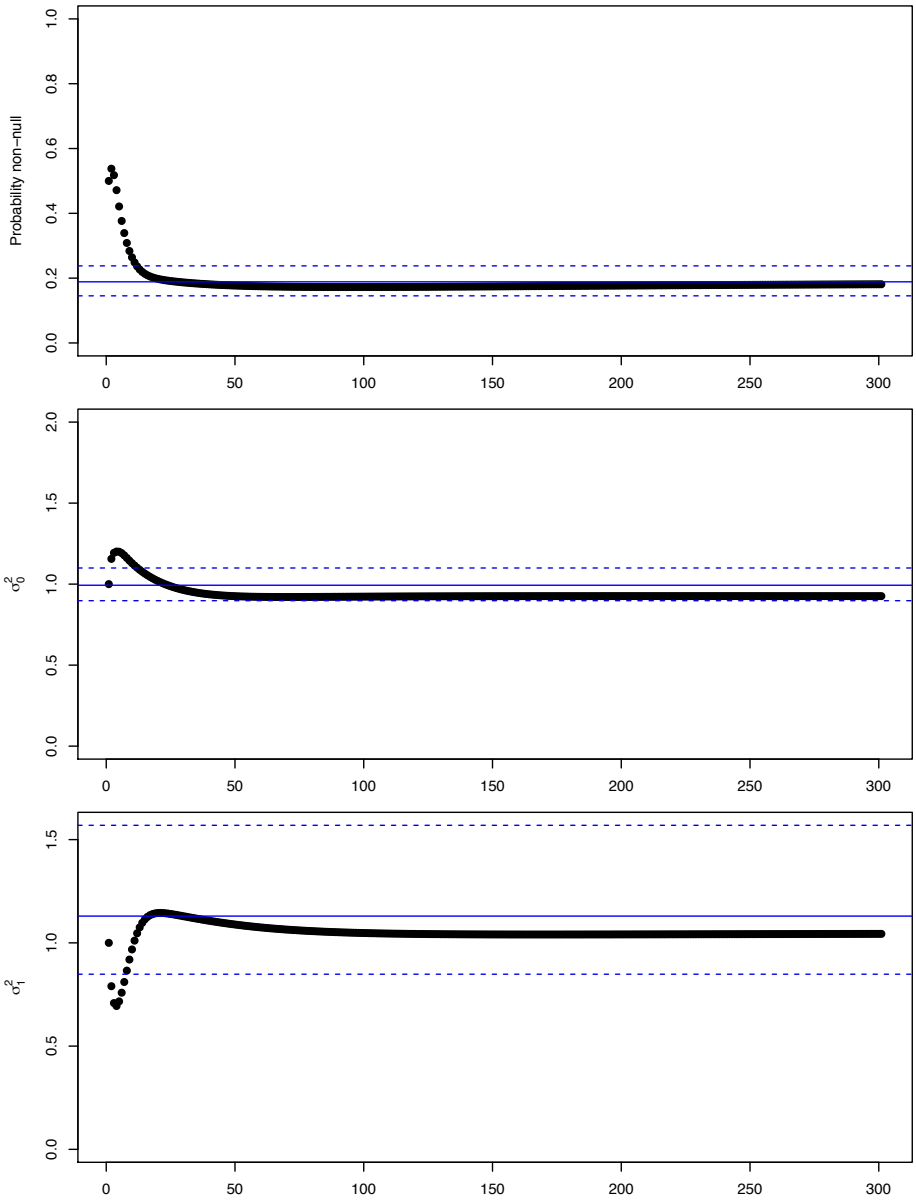
Estimation of parameters for CPTAC Yeast synthetic data.



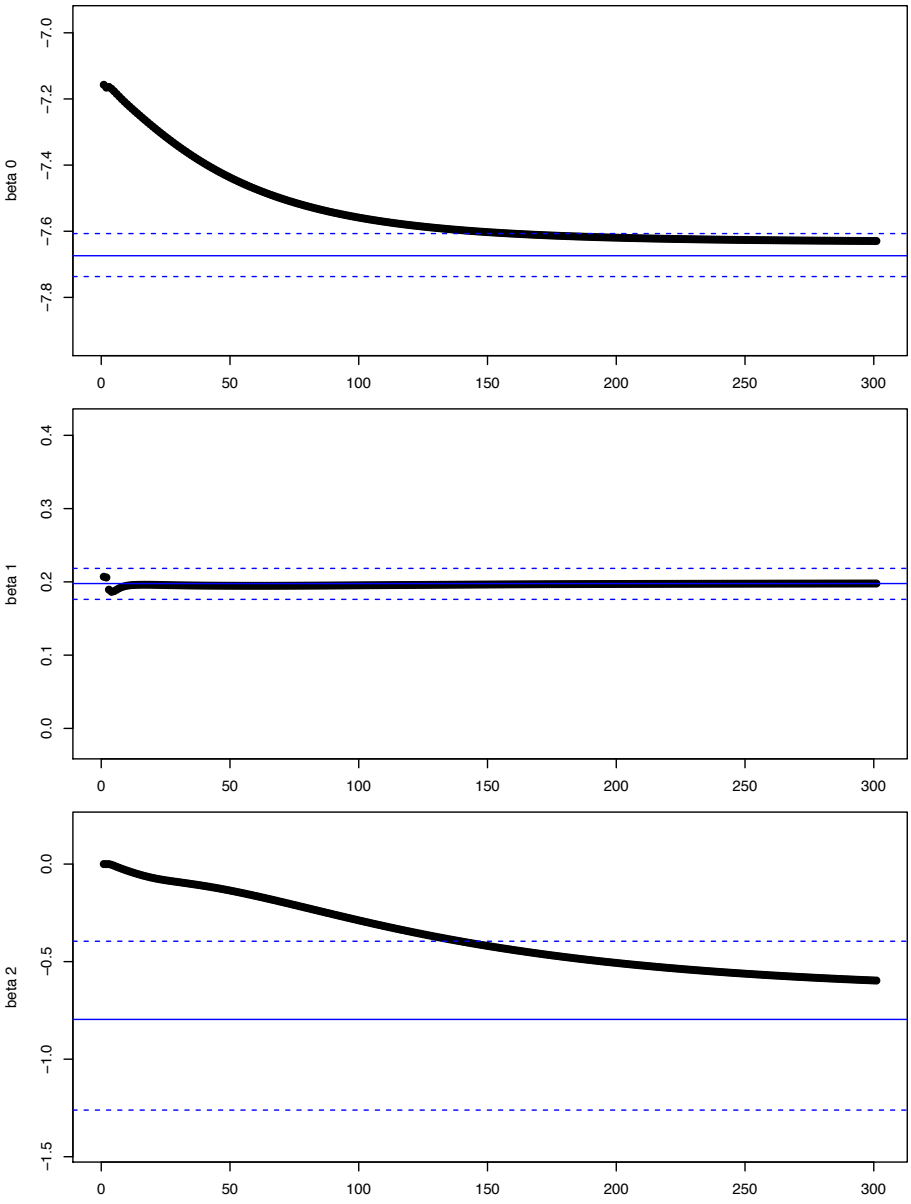

Probability non−null
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
σ_0^2
3.0
2.5
2.0
1.5
1.0
0.5
0.0
σ_1^2

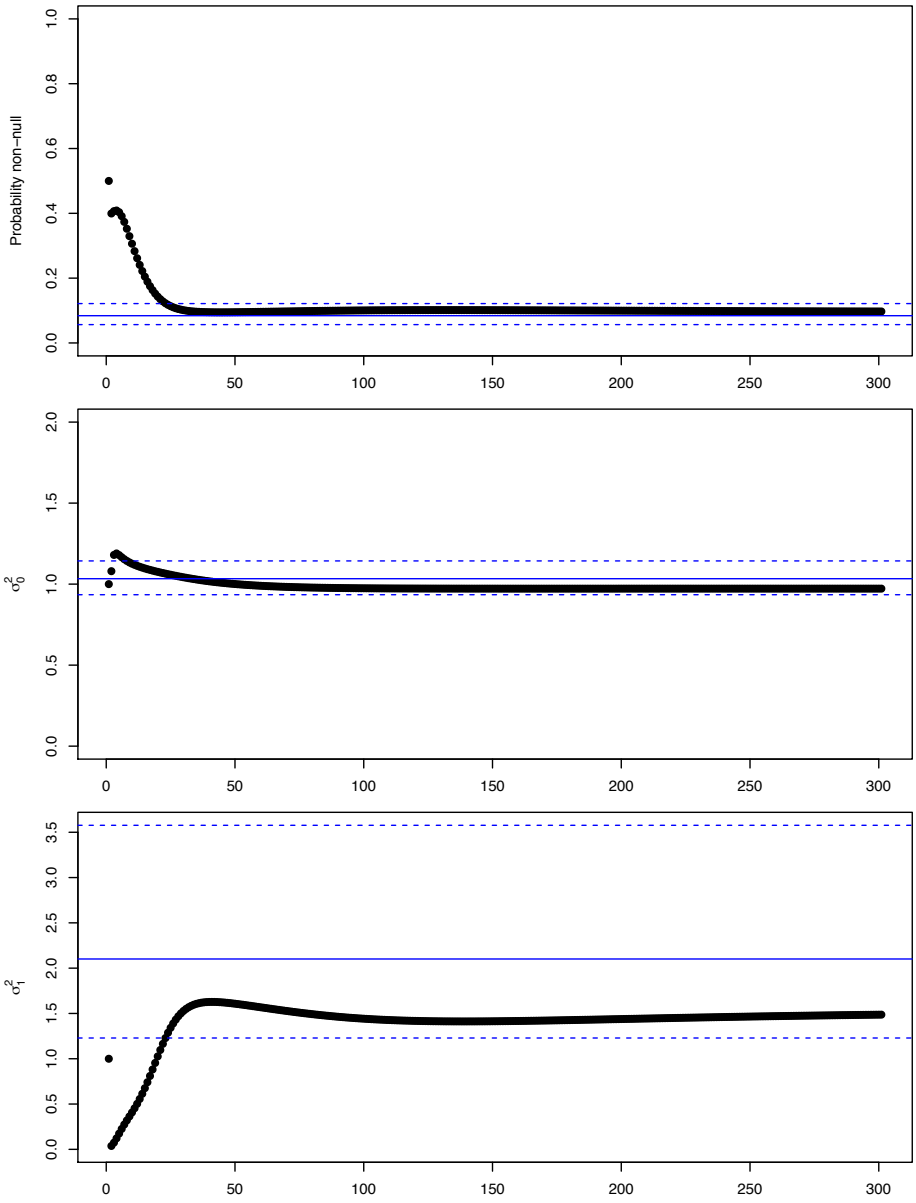
Estimation of parameters for 2 fold increase data.




Probability non−null
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
σ_0^2
2.0
1.5
1.0
0.5
0.0
σ_1^2
1.5
1.0
0.5
0.0

Estimation of parameters for 2 fold decrease data.




Probability non−null
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
250
300
σ_0^2
0.0
0.5
1.0
1.5
2.0
σ_1^2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

APPENDIX B

SNIPRE CODE

B.1 SnIPRE and B SnIPRE Function Definitions

Functions for B SnIPRE (Bayesian implementation)

```
# G(n) from equation 1.4 in dissertation

Gnint <- function(s,n=30){
int <- 0
  f <- function(x,g){(1-x)^(n-1) * ( 1- exp(-2*g*x))/(2*g*x)}
for(i in 1:length(s)){
  int[i] <- integrate(f,g= s[i], 0,1)$value}
return(int)
}

# F(n) from equation 1.3 in dissertation

Fnint <- function(s, n = 30){
int <- 0
  f <- function(x,g){(1-x^(n) - (1-x)^(n))/(1-x)* ( 1- exp(-2*g*x))/(2*g*x)}
for(i in 1:length(s)){
  int[i] <- integrate(f,g= s[i], 0,1)$value}
return(int)
}

# solution to "func" is the selection effect

func <- function(g, ef, tau, n=30, m =1){
  L.n <- sum(1/(c(1:(n-1))))
  f <- ef - log((tau +Gnint(g,m)+Gnint(g,n))/(Fnint(g, n))*((L.n)/(tau+1/m+1/n)))
return(f)
}

# uses the unit root function to find root of "func"

LLest2gamm <- function(effect,tau.est = rep(10, length(effect)),
                       n = rep(30, length(effect)),m =rep(1, length(effect))){
gest <- 0
neg <- which(tau.est < 0)
tau.est[neg] <- 0
for(i in 1:length(effect)){
    worked <- try(gest[i] <- uniroot(func, lower=-350, upper=300,
                                     ef=effect[i],tau=tau.est[i], n=n[i], m = m[i])$root)
    if (class(worked)=="try-error") gest[i] <- NA
                }
return(gest)
}
```

```
# estimate mutation rate

theta <- function(gene.effect.est, m, n){
  th <- exp(gene.effect.est)/(Ln(m)+Ln(n))
  return(th)
}

# function calls estimation of mutation rate, and estimates tau
tau.theta <- function(gene.effect.est,f.effect.est,
                      m=rep(1, length(gene.effect.est)),
                      n=rep(30, length(gene.effect.est))){
  th <- theta(gene.effect.est, m, n)
  t <- exp(gene.effect.est+f.effect.est)/th  - 1/m - 1/n
  return(list(tau.est = t, theta.est=th))
}


# root of function is estimate f (g = estimate of gamma)
funcf <- function(frac, g, r.eff, npop=30, nout=1){
  denom <- Ln(npop)+Ln(nout)
  f <- r.eff - log(frac*(2*g)/(1-exp(-2*g))*(Fnint(g,npop)+Fnint(g, nout))/denom)
  return(f)
}

#functions that find roots of funcf
frac.est <- function(r.eff, g, npop = rep(30, length(r.eff)),
                     nout = rep(1, length(r.eff))){
  frac <- 0
  for(i in 1:length(r.eff)){
    frac[i] <-  uniroot(funcf, lower = 0, upper = 100,
                        r.eff = r.eff[i],g = g[i], npop=npop[i], nout=nout[i])$root
  }
  return(frac)
}

# L(n) from equation 1.2 in disseration
Ln <- function(n){
  v <- 0
  t <- 0
  for(i in 1:length(n)){
  v <- c(1:(n[i]-1))
  t[i] <- sum(1/v)
  if(n[i]==1) t[i] <- 0}
  return(t)
}

# calculates the probability of being "negative"

pn <- function(vec){
  negprop <- length(which(vec < 0))/length(vec)
  return(negprop)}
```

Main function call for B SnIPRE

```
BSnIPRE <- function(n,data.mcmc,name, npop=rep(30, n), nout=rep(1,n)){
  BRF <- data.mcmc[,which(varnames(data.mcmc) == "m[4]")]
  BR <- data.mcmc[,which(varnames(data.mcmc) == "m[2]")]
  BF <- data.mcmc[,which(varnames(data.mcmc)=="m[3]")]
  B <- data.mcmc[,which(varnames(data.mcmc)=="m[1]")]
  BGst <- which(varnames(data.mcmc) == "BG[1,1]")
  BG <-data.mcmc[, c(BGst:(BGst+4*n-1))]

  RF <- c((1:n)*4)
  R <- RF - 2
  F <- RF - 1
  G <- RF - 3

  gene.effect1 <- BG[,G,][[1]]
  gene.effect2 <- BG[,G,][[2]]
  gene.effect3 <- BG[,G,][[3]]

  gene.effect <- mcmc.list(gene.effect1, gene.effect2, gene.effect3)

  sel.effect1 <- BG[,RF,][[1]]
  sel.effect2 <- BG[,RF,][[2]]
  sel.effect3 <- BG[,RF,][[3]]

  sel.effect <- mcmc.list(sel.effect1, sel.effect2, sel.effect3)

  r.effect1 <- BG[,R,][[1]]
  r.effect2 <- BG[,R,][[2]]
  r.effect3 <- BG[,R,][[3]]

  r.effect <- mcmc.list(r.effect1, r.effect2, r.effect3)

  f.effect1 <- BG[,F,][[1]]
  f.effect2 <- BG[,F,][[2]]
  f.effect3 <- BG[,F,][[3]]

  f.effect <- mcmc.list(f.effect1, f.effect2, f.effect3)


  sel.probs <- apply(as.matrix(sel.effect), 2, quantile, probs = c(.025, .5, .975))
  pneg <- apply(as.matrix(sel.effect), 2, pn)
  r.probs <- apply(as.matrix(r.effect), 2, quantile, probs = c(.025, .5, .975))
                                        # Selection Effect Estimate/Classification
  name$BglmmGEN.class <- 0
  name$BglmmGEN.est <- sel.probs[2,]
  name$BglmmGEN.lbound <- sel.probs[1,]
  name$BglmmGEN.ubound <- sel.probs[3,]
  name$BglmmGEN.class = "neut"
  name$BglmmGEN.class[which(name$BglmmGEN.lbound>0)] = "pos"
  name$BglmmGEN.class[which(name$BglmmGEN.ubound<0)] = "neg"
  name$BglmmGEN.pneg <- pneg

                                        # Replacement Effect Estimate/Classification
```

```
  name$BglmmGEN.Rest <- r.probs[2,]
  name$BglmmGEN.Rlbound <- r.probs[1,]
  name$BglmmGEN.Rubound <- r.probs[3,]

  name$BglmmGEN.Rclass <- 0
  name$BglmmGEN.Rclass = "neut"
  name$BglmmGEN.Rclass[which(name$BglmmGEN.Rlbound>0)] = "pos"
  name$BglmmGEN.Rclass[which(name$BglmmGEN.Rubound<0)] = "neg"

                                        # gene effect estimate
 probs.gene <- apply(as.matrix(gene.effect), 2, quantile,
                     probs = c(.025,.5,.975))
                                        # fixed effect estimate
 probs.fix <- apply(as.matrix(f.effect), 2, quantile, probs = c(.025,.5,.975))
 gene.effect.est <- probs.gene[2,]
 f.effect.est <- probs.fix[2,]
                                        # estimate tau & theta
 params <- tau.theta(gene.effect.est,f.effect.est, npop, nout)
 name$BglmmGEN.tau = params$tau.est
 name$BglmmGEN.theta = params$theta.est
 name$BglmmGEN.gamm[name$BglmmGEN.est>-4.35] <-  LLest2gamm(
                   name$BglmmGEN.est[name$BglmmGEN.est>-4.35],
                   name$BglmmGEN.tau[name$BglmmGEN.est>-4.35],
                   n = npop, m = nout)
na.set <- which(is.na(name$BglmmGEN.gamm) == TRUE)
good.set <- which(name$BglmmGEN.est > - 4.4)
use <- setdiff(good.set, na.set)
name$Bayesian.f <- NA
name$Bayesian.f.lb <- NA
name$Bayesian.f.ub <- NA
g.ests <- name$BglmmGEN.gamm
g.zeros <- which(name$BglmmGEN.class == "neut")
g.ests[g.zeros] <- .000001
name$Bayesian.f[use] <- frac.est(name$BglmmGEN.Rest[use], g.ests[use],
                                       npop[use], nout[use])
name$Bayesian.f.lb[use] <-frac.est(name$BglmmGEN.Rlbound[use],
                                   g.ests[use],  npop[use],nout[use])
name$Bayesian.f.ub[use] <-frac.est(name$BglmmGEN.Rubound[use],
                                   g.ests[use], npop[use],nout[use])
  name$Bayesian.f.class <- "neut"
  name$Bayesian.f.class[which(name$Bayesian.f.ub<1)] = "neg"
  name$Bayesian.f.class[which(name$Bayesian.f.lb>1)] = "pos"
return(list(new.dataset = name, sel.effect = sel.effect, rep.effect = r.effect,
             fix.effect = f.effect, gene.effect = gene.effect) )
}
```

Functions for SnIPRE (empirical Bayes implementation)

```
# G(n) from equation 1.4 in dissertation

Gnint <- function(s,n=30){
int <- 0
  f <- function(x,g){(1-x)^(n-1) * ( 1- exp(-2*g*x))/(2*g*x)}
for(i in 1:length(s)){
  int[i] <- integrate(f,g= s[i], 0,1)$value}
return(int)
}

# F(n) from equation 1.3 in dissertation

Fnint <- function(s, n=30){
int <- 0
  f <- function(x,g){(1-x^(n) - (1-x)^(n))/(1-x)* ( 1- exp(-2*g*x))/(2*g*x)}
for(i in 1:length(s)){
  int[i] <- integrate(f,g= s[i], 0,1)$value}
return(int)
}

# root of function is estimate f (g = estimate of gamma)

funcf <- function(frac, g, r.eff, npop=30, nout=1){
  denom <- Ln(npop)+Ln(nout)
  f <- r.eff - log(frac*(2*g)/(1-exp(-2*g))*(Fnint(g,npop)+Fnint(g, nout))/denom)
  return(f)
}

#functions that find roots of funcf

frac.est <- function(r.eff, g, npop = rep(30, length(r.eff)),
                     nout = rep(1, length(r.eff))){
  frac <- 0
  for(i in 1:length(r.eff)){
    frac[i] <-  uniroot(funcf, lower = 0, upper = 100, r.eff = r.eff[i],g = g[i],
                                  npop=npop[i], nout=nout[i])$root
  }
  return(frac)
}

# L(n) from equation 1.2 in disseration

Ln <- function(n){
  v <- 0
  t <- 0
  for(i in 1:length(n)){
  v <- c(1:(n[i]-1))
  t[i] <- sum(1/v)
  if(n[i]==1) t[i] <- 0}
  return(t)
}
```

```
# estimate mutation rate

theta <- function(beta, betaG, m, n){
  th <- exp(beta+betaG)/(Ln(m)+Ln(n))
  return(th)
}

# solution to "func" is the selection effect

func <- function(g, ef, tau, n=30,m =1){
    L.n <- sum(1/(c(1:(n-1))))
  f <- ef - log((tau +Gnint(g,m)+Gnint(g,n))/(Fnint(g, n))*((L.n)/(tau+1/m+1/n)))
return(f)
}

# uses the unit root function to find root of "func"

LLest2gamm <- function(effect,tau.est = rep(10, length(effect)),
                       n = rep(30, length(effect)), m = rep(1, length(effect))){
gest <- 0
neg <- which(tau.est < 0)
tau.est[neg] <- 0
for(i in 1:length(effect)){
    worked <- try(gest[i] <- uniroot(func, lower=-350, upper=300,
                                     ef=effect[i],tau=tau.est[i],
                                     n=n[i],m = m[i])$root)
    if (class(worked)=="try-error") gest[i] <- NA
                }
return(gest)
}

# function calls estimation of mutation rate, and estimates tau

tau.theta <- function(beta, betaG, betaF, betaFG, m=1, n=30){
  th <- theta(beta, betaG, m, n)
  t <- exp(beta+betaG +betaF+betaFG)/th  - 1/m - 1/n
  return(list(tau.est = t, theta.est=th))
}
```

Main function call for SnIPRE

```
SnIPRE <-function(name, npop=rep(30, dim(name)[1]), nout=(rep(1, dim(name)[1]))){
 data <- name
 PS <- data$PS
 PR <- data$PR
 FR <- data$FR
 FS <- data$FS

 n <- length(FS)
 TS <- data$Total.Syn
 TR <- data$Total.Non.Syn

 Ivec <- matrix(1, nrow = n)  # makes one vector of appropriate size subset
 d.mu <-as.numeric( matrix(c(PS,PR,FS,FR), ncol =1))
```

```
 d.replacement <- as.numeric(c(0,1,0,1)%x%Ivec)
 d.fixed <- as.numeric(c(0,0,1,1)%x%Ivec)
 d.gene <- as.vector(rep(1,4)%x%c(1:n))
 d.TS <- as.vector(rep(1,4)%x%c(TS))
 d.TR <- as.vector(rep(1,4)%x%c(TR))

 count <- as.vector(d.mu)
 R <- as.vector(d.replacement)
 F <- as.vector(d.fixed)
 G <- as.vector(d.gene)
 RF <- as.vector(R*F)
 TR <- as.vector(d.TR)
 TS <- as.vector(d.TS)

 modGEN <- glmer(count~ 1+ R + F + RF +(1+R+F+RF|G),offset =  log(TS*(1-R)+TR*R), family = poisson)
                                        # sel effect
 se.RFG = se.ranef(modGEN)$G[,4]
 re.RFG = ranef(modGEN)$G$RF
 lbound <- fixef(modGEN)[4]+re.RFG - 1.96*se.RFG
 ubound <- fixef(modGEN)[4]+re.RFG + 1.96*se.RFG

 beta <- fixef(modGEN)[1]
 betaG <- ranef(modGEN)$G[,1]
 betaF <- fixef(modGEN)[3]
 betaFG <- ranef(modGEN)$G[,3]

 name$glmmGEN.lbound <- lbound
 name$glmmGEN.ubound <- ubound

 negC <- which(ubound<=0)
 posC <- which(lbound>=0)

 name$glmmGEN.class <- "neut"
 name$glmmGEN.class[negC] <- "neg"
 name$glmmGEN.class[posC] <- "pos"
 name$glmmGEN.est <- fixef(modGEN)[4]+re.RFG
                                        # replacement effect

re.RG = ranef(modGEN)$G$R
se.RG = se.ranef(modGEN)$G[,2]
Rlbound <- fixef(modGEN)[2]+re.RG - 1.96*se.RG
Rubound <- fixef(modGEN)[2]+re.RG + 1.96*se.RG

name$glmmGEN.Rlbound <- Rlbound
name$glmmGEN.Rubound <- Rubound

negR <- which(Rubound<=0)
posR <- which(Rlbound>=0)
params <- tau.theta(beta, betaG,betaF,betaFG, npop, nout)
name$glmmGEN.tau = params$tau.est
name$glmmGEN.theta = params$theta.est
name$glmmGEN.Rclass <- "neut"
name$glmmGEN.Rclass[negR] <- "neg"
name$glmmGEN.Rclass[posR] <- "pos"
name$glmmGEN.Rest <- fixef(modGEN)[2]+ranef(modGEN)$G[,2]
```

```
name$glmmGEN.gamm[name$glmmGEN.est<=-4.4] <- -Inf
name$glmmGEN.gamm[name$glmmGEN.est>-4.4] <-  LLest2gamm(name$glmmGEN.est[name$glmmGEN.est>-4.4],
                                                        name$glmmGEN.tau[name$glmmGEN.est>-4.4], n = npo
na.set <- which(is.na(name$glmmGEN.gamm) == TRUE)
good.set <- which(name$glmmGEN.est > - 4.4)
use <- setdiff(good.set, na.set)
name$f <- NA
name$f.lb <- NA
name$f.ub <- NA
g.ests <- name$glmmGEN.gamm
g.zeros <- which(name$glmmGEN.class == "neut")
g.ests[g.zeros] <- .000001
name$f[use] <- frac.est(name$glmmGEN.Rest[use], g.ests[use],
                                          npop[use], nout[use])
name$f.lb[use] <-frac.est(name$glmmGEN.Rlbound[use], g.ests[use], npop[use],nout[use])
name$f.ub[use] <-frac.est(name$glmmGEN.Rubound[use], g.ests[use], npop[use],nout[use])
name$f.class <- "neut"
name$f.class[which(name$f.ub <1)] <- "neg"
name$f.class[which(name$f.lb >1)] <- "pos"
return(list(new.dataset = name, model = modGEN))
}


save.image("analysis_SnIPRE.Rdata")
```

B.2 WinBUGS model

Note: this model specification will also work in OpenBUGS

```
model{
  for(i in 1:4*n)
    {
      count[i] ~ dpois(mu[i])
      log(mu[i]) <- log(TS[i])*(1-R[i])+log(TR[i])*R[i]+inprod(BG[G[i],],x[i,])
      res[i]<-count[i]-mu[i]
    }
    for(i in 1:n){
   BG[i,1:4] ~ dmnorm(m[1:4], T[1:4,1:4])
  }

   v <-10
  T[1:4,1:4] ~ dwish(S[1:4,1:4], v)

 m[1]~dnorm(0, .001)
 m[2]~ dnorm(0, .001)
 m[3] ~ dnorm(0, .001)
 m[4]~dnorm(0, .001)
}
```

B.3 Script for analysis

```
library(R2WinBUGS)
library(mvtnorm)
library(MASS)
library(lme4)
library(GenKern)
library(arm)
library(gdata)

# data = read.table("data.txt")
# data should have variables: PS, PR, FR, FS, Total.Syn, Total.Non.Syn, npop, nout

##############################################################################
                                        # Fit B SnIPRE
##############################################################################
# define the following variables from the data

PS <- data$PS
PR <- data$PR
FR <- data$FR
FS <- data$FS

n <- length(FS)
```

```
TS <- data$Total.Syn
TR <- data$Total.Non.Syn

# format data
Ivec <- matrix(1, nrow = n)  # makes one vector of appropriate size subset
d.mu <-as.numeric( matrix(c(PS,PR,FS,FR), ncol =1))
d.replacement <- as.numeric(c(0,1,0,1)%x%Ivec)
d.fixed <- as.numeric(c(0,0,1,1)%x%Ivec)
d.gene <- as.vector(rep(1,4)%x%c(1:n))
d.TS <- as.vector(rep(1,4)%x%c(TS))
d.TR <- as.vector(rep(1,4)%x%c(TR))

     # respones:
count <- as.vector(d.mu)
     #design for x matrix
R <- as.vector(d.replacement)
F <- as.vector(d.fixed)
G <- as.vector(d.gene)
RF <- as.vector(R*F)
x <- matrix(c(rep(1,4*n),R,F,RF), nrow = 4*n)
     # offsets
TR <- as.vector(d.TR)
TS <- as.vector(d.TS)
v = 10
iS = diag(4)
S <- solve(iS)/v  #prior value

wb.data <- list("count","R","n","G", "TR", "TS","x","S")  # data to be read in

      # function to create initial values
inits = function(){list(
  BG = array(mvrnorm(n,c(0,0,0,0),iS),c(n,4)),
  T = solve(iS),
  m = c(0,0,0,0)
 )}

setwd("put my results here")
library(R2WinBUGS)

mcmc.loc <- bugs(wb.data,inits,  n.chains = 3,
                    model.file = "path to SNIPRE_model.bug",
                     parameters = c("m","BG","T"),
                   n.iter = 10000, n.thin=5, n.burnin = 5000, debug = FALSE,codaPkg=TRUE,
                    DIC = FALSE, save.history = FALSE)
      # if you type mcmc.loc, now it should have the location of your coda files
      # read in coda files
mcmc.res <- read.bugs(c("coda1.txt", "coda2.txt", "coda3.txt"), quiet =TRUE)

                                        #Example Diagnostics:
      # for n genes, 4*n+17:20 is B, BR, BF, BRF (m[1:4]); 4*n +1:16 is T[1,1]
BG= 1:(4*n)
m = 4*n+17:20
T = 4*n+1:16
plot(mcmc.res[,m,])
plot(mcmc.res[,BG[1:4],])
```

```
plot(mcmc.res[,T[1:4],])
gelman.diag(mcmc.res[,T,])

load("BSnIPRE.Rdata")  #function values should be saved here
res <- BSnIPRE(dim(data)[1], mcmc.res, data, npop, nout)

data <- res$new.dataset
write.table(data, "BSnIPRE_results.txt", row.names = FALSE)


#########################################################################
                                        # fit SnIPRE
#########################################################################
load("SNIPRE.Rdata") # function values should be saved here
res <- SnIPRE(data, npop, nout)

data <- res$new.dataset
write.table(data, "SnIPRE_results.txt", row.names = FALSE)
SnIPRE.mod = dist1.res$model
save(SnIPRE.mod, file = "SnIPRE_model")
```

APPENDIX C

PROWLRE CODE

C.1 Function definitions

```
# H function: returns H^{-1}
# Calculates both the null and nonnull H

 H <- function(mu0p, mu1p, D, n){
  znull <- c(1,0)
  znn <- c(1,1)
  Dinv = solve(D)
     H0inv= sum(mu0p)*znull%*%t(znull)+Dinv
   # H1
    H1invA = sum(mu1p[1:(n/2)])*znull%*%t(znull)
    H1invB = sum(mu1p[((n/2)+1):n])*znn%*%t(znn)
    H1inv <- H1invA+H1invB+Dinv
  return(list(H0inv = H0inv, H1inv = H1inv))
}

# the h() from equation 4.5 (dissertation)
# null z = c(1,0);  nonnull z = c(1,1)
h <- function(y, mu, D, b, z){
  Dinv = solve(D)
  re <- b*z
  w = sum(y*log(mu) - mu) - 1/2*t(re)%*%Dinv%*%(re)
  return(w)
}

#yp: the pth proteins counts;
#mu0p: the fitted means under null;
#mu1p: the fitted means under nonnull;
#D: covariance of random effects
#bnull and bnn: random effects in null and nonnull models
#n: number replicates for each protein (n/2 per treatment)

#f in equation 4.6 from disseration
f <- function(yp,mu0p, mu1p, D, bnull,bnn,n){
    z0 = c(1,0)
    z1 = c(1,1)
    tildeH = H(mu0p, mu1p, D,n)  # calculates H0inv and H1inv
    H0= solve(tildeH$H0inv)  #get H0
    H1= solve(tildeH$H1inv)  #get H1
    h0 = h(yp, mu0p, D, bnull,z0)  # get h for under null
    h1 = h(yp, mu1p, D, bnn,z1)  # get h for under nonnull
    lf0 =h0-sum(lfactorial(yp))+log(det(2*pi*H0)^(1/2))   # log of the approximation
    lf1 = h1-sum(lfactorial(yp))+log(det(2*pi*H1)^(1/2))
    return(list(lf0 =lf0, lf1 = lf1, dH0 = det(H0), dH1 = det(H1), h0 = h0,h1 = h1))
  }
```

```
#function to subtract off old random effects and add in new random effects
mk.mu <- function(mod.mu, b0, b1 = 0, b0i, b1i = 0){
      Gmat <- matrix(G, nrow = p, ncol = n)
      lm <- log(mod.mu) - b0 - Gmat*b1 +b0i +Gmat*b1i
      m <- exp(lm)
      return(m)}


blups.update <- function(D, mu.null, mu.nonnull,
                         bnull.last = cbind(rep(0,p),rep(0,p)),
                         bnn.last = cbind(rep(0,p),rep(0,p)),
                         rm = FALSE, p = p, S = S, Gmat = Gmat, Ym = Ym){

    f0b0 <- matrix(0, ncol = S+1, nrow = p)# null model blups
    f1b0 <- matrix(0, ncol = S+1, nrow = p)# nonnull model blups
    f1b1 <- matrix(0, ncol = S+1, nrow = p)
    f0b0[,1] <- bnull.last[,1]
    f1b0[,1] <- bnn.last[,1]#b0
    f1b1[,1] <- bnn.last[,2]#b1
    s = 0
    varb0 <- D[1]
    varb1 <- D[4]
    if(!rm){
      oldb0null <- 0
      oldb0nn <- 0
      oldb1nn <- 0
    }
    if(rm){
      oldb0null <- bnull.last[,1]
      oldb0nn <- bnn.last[,1]
      oldb1nn <- bnn.last[,2]
    }

    while(s<S){
      s=s+1
                                        #For Null model
        mnull <- mk.mu(mu.null, b0 = oldb0null, b0i = f0b0[,s])
      f0b0[,s+1] = f0b0[,s]- (apply((Ym -mnull),1,sum) - f0b0[,s]/varb0)/
        (apply(-mnull,1,sum) - 1/varb0)
                                        #For NONNULL model
     mnn <- mk.mu(mu.nonnull, b0 = oldb0nn,b1 = oldb1nn,b0i = f1b0[,s],b1i = f1b1[,s])
      j11 = apply(- mnn,1,sum) - 1/varb0
      j12 = apply(- mnn*Gmat,1,sum)
      j21 = j12
      j22 = apply(- mnn*Gmat,1,sum) - 1/varb1
      F1 = (apply((Ym - mnn),1,sum) - f1b0[,s]/varb0)
      F2 = (apply((Ym - mnn)*Gmat,1,sum) - f1b1[,s]/varb1)
     for(q in 1:p){
        a = matrix(c(j11[q],j12[q], j21[q], j22[q]), nrow =2)
        b = matrix(c(-F1[q],-F2[q]), nrow = 2)
        worked <- try(diff <- solve(a,b), silent = TRUE)
        if(class(worked) == "try-error"){ diff = c(.5,.5)}
        f1b0[q,s+1] <- f1b0[q,s]+diff[1]
        f1b1[q,s+1] <- f1b1[q,s]+diff[2]
        if(max(abs(diff)) >100){ f1b0[q,s+1] <- 0; f1b1[q,s+1] <- 0}
```

```
      }
    }

ddntwrk0<- which(abs(f0b0[,s+1]-f0b0[,s])>.01)
ddntwrk1 <- union(ddntwrk0,union(which(abs((f1b0[,s+1]-f1b0[,s]))>.01),which(abs((f1b1[,s+1]-f1b1[,s]))>

   bnull <- cbind(f0b0[,S+1],rep(0,p))
   bnull[ddntwrk0,1] <- 0
   bnn <- cbind(f1b0[,S+1],f1b1[,S+1])
    bnn[ddntwrk1,] <- c(0,0)
    mnull <- mk.mu(mu.null, b0 = oldb0null, b0i = f0b0[,S+1])
    mnn <- mk.mu(mu.nonnull, b0 = oldb0nn,b1 = oldb1nn,b0i = f1b0[,S+1],b1i = f1b1[,S+1])

    return(list(bnull = bnull, bnn= bnn, mnull = mnull, mnn = mnn, ddntwrk1=ddntwrk1))
  }

cov0.func <- function(bnull, bnn, mu.null, mu.nonnull, uEIp, sig0.init, sig2.1,n){
  n <- dim(mu.null)[2]
  p0k <- (1-uEIp)
  p1k <- uEIp
  sig2.0 <- sig0.init
  m0 <- apply(mu.null,1,sum)
  m1a <- apply(mu.nonnull[,1:(n/2)],1,sum)
  m1b <- apply(mu.nonnull[,(n/2+1):n],1,sum)
  A0 <- (m0*sig2.0^2+sig2.0)^(-1)
  A1 <- 1/sig2.0^2*((m1b+1/sig2.1)/((m1a+m1b+1/sig2.0)*(m1b+1/sig2.1)- m1b^2))

  dA0 <- -(m0*sig2.0^2 +sig2.0)^(-2)*(2*m0*sig2.0 +1)
  dA1 <- 1/sig2.0^3*(-2*(m1b+1/sig2.1)/((m1a+m1b+1/sig2.0)*(m1b + 1/sig2.1)-m1b^2)+
                     (m1b+1/sig2.1)^2/((m1a+m1b+1/sig2.0)*(m1b + 1/sig2.1)-m1b^2)^2)

  cov.prime <-  sum(p0k*(bnull[,1]^2/sig2.0^2 - (1/sig2.0) +A0)+
                    p1k*(bnn[,1]^2/sig2.0^2 - (1/sig2.0) +A1))
  cov.dprime <- sum(p0k*(-2*(bnull[,1]^2)/sig2.0^3 + (1/sig2.0)^2 + dA0)+
                    p1k*(-2*(bnn[,1]^2)/sig2.0^3 + (1/sig2.0)^2 + dA1))

  sig2.0.up <- max(sig2.0 - cov.prime/cov.dprime,.001)
  if(cov.dprime > 0){
    sig2.0.up <- sig2.0
    print("sig0 not max")
  }
  return(sig2.0.up)
}

cov1.func <- function(bnull, bnn, mu.null, mu.nonnull, uEIp, sig2.0, sig1.init,n){
  n <- dim(mu.null)[2]
  p0k <- (1-uEIp)
  p1k <- uEIp
  sig2.1 <- sig1.init
  m0 <- apply(mu.null,1,sum)
  m1a <- apply(mu.nonnull[,1:(n/2)],1,sum)
  m1b <- apply(mu.nonnull[,(n/2+1):n],1,sum)
  A0 <- 1/sig2.1
  A1 <- 1/sig2.1^2*((m1a+m1b+1/sig2.0)/((m1a+m1b+1/sig2.0)*(m1b+1/sig2.1)- m1b^2))
```

```
  dA0 <- -1/sig2.1^2
  dA1 <- 1/sig2.1^3*(-2*(m1a+m1b+1/sig2.0)/((m1a+m1b+1/sig2.0)*(m1b+1/sig2.1)- m1b^2)+
                     (m1a+m1b+1/sig2.0)^2/((m1a+m1b+1/sig2.0)*(m1b+1/sig2.1)- m1b^2)^2)

  cov.prime <- sum(p0k*(bnull[,2]^2/sig2.1^2 - (1/sig2.1) +A0)+
                   p1k*(bnn[,2]^2/sig2.1^2 - (1/sig2.1) +A1))
  cov.dprime <- sum(p0k*(-2*(bnull[,2]^2)/sig2.1^3 + (1/sig2.1)^2 + dA0)+
                    p1k*(-2*(bnn[,2]^2)/sig2.1^3 + (1/sig2.1)^2 + dA1))

  sig2.1.up <- max(sig2.1 - cov.prime/cov.dprime, .001)
  if(cov.dprime > 0){
    sig2.1.up <- sig2.1
    print("sig 1 not max")
  }

  return(sig2.1.up)
}



NR.cov <- function(bnull, bnn, mu.null, mu.nonnull, uEIp, sig0.init, sig1.init, niter,n){
  sig0 <- sig0.init
  sig1 <- sig1.init
  for(i in 1:niter){
    sig0[i+1] <- cov0.func(bnull, bnn, mu.null, mu.nonnull, uEIp, sig0[i], sig1[i],n)
    sig1[i+1] <- cov1.func(bnull, bnn, mu.null, mu.nonnull, uEIp, sig0[i+1], sig1[i],n)
  }
  return(list(sig2.0 =sig0, sig2.1 =sig1))
}


save.image("/Users/Kirsten/EM_functions.Rdata")
```

Main Function Call:

```
J: number of iterations
p1: initial probability of nonnull (usually .5)
EI: initial probability of nonnull for particular protein (usually .5)
Y: response counts (matrix p x n)
G: treatment group (matrix pxn of 0s and 1s)
O: offset (matrix p x n)
n: number of replicates per protein (balanced design set up, assumes
   n/2 pergroup
p: number of proteins
varb0, varb1: initial or fixed values for variance
FIXVAR: is variance to be estimated? FALSE indicates yes

EM <- function(J, p1, EI, Y, G, O, n, p, varb0 = 1, varb1 = 1,FIXVAR = FALSE){
  Ym <- matrix(as.vector(Y), ncol = n, byrow = FALSE)
  D <- diag(c(varb0,varb1))
  Yfit <- c(Y,Y)
  Gfit <- c(G,G)
  I <- c(rep(0,n*p),rep(1,n*p))
```

```
  offs1 = c(O,O)

  EIp <- matrix(0,ncol = J+1, nrow = p)
  EIp[,1] <- EI
  fixeff <- matrix(0, nrow = 5, ncol =J+1)
  Gmat <- matrix(G, nrow = p, ncol = n)

                                          # Step 1
                                        # fit model
  mod.glm <- glm(Yfit~1+Gfit+Gfit:I, offset = offs1, family = poisson(link = "log"),
                   weights = c(rep((1-EIp[,1]),n), rep(EIp[,1],n)) )

  print(D)
     fixeff[,1] <- c(mod.glm$coef, varb0,varb1)
  mu.null <- matrix(mod.glm$fitted[1:(n*p)], byrow =FALSE, ncol = n)
  mu.nonnull <- matrix(mod.glm$fitted[(n*p+1):(2*n*p)], byrow =FALSE, ncol = n)
for(j in 1:J){
                                        #Use NR to update blups
  if(j ==1) {
    bup <- blups.update(D,mu.null, mu.nonnull, S = 50, p = p, Gmat = Gmat, Ym = Ym)
  }
  if(j==2){
      bup <- blups.update(D,mu.null, mu.nonnull,bnull.last = bnull,
                          bnn.last = bnn, S = 50, p = p, rm = TRUE, Gmat = Gmat, Ym = Ym)
  }
  if(j>2){
    bup <- blups.update(D,mu.null, mu.nonnull,bnull.last = bnull,
                        bnn.last = bnn, S = 10, p = p, rm = TRUE, Gmat = Gmat, Ym = Ym)
      }

  bnull <- bup$bnull
  bnn <- bup$bnn
                                        #expected values (w/ new blups)
  m.null <- bup$mnull
  m.nonnull <- bup$mnn
  lf1 = 0  #log laplace nonnull
  lf0 = 0  #log laplace null
                                        #Laplace approx EIp
  for(i in 1:p){
   worked <- try(r <- f(Ym[i,], m.null[i,], m.nonnull[i,], D,bnull[i,], bnn[i,]))
        if(class(worked) == "try-error"){ r$lf1 = NA; r$lf0= NA}
      lf1[i] <- r$lf1
      lf0[i] <- r$lf0
    }
         EIp[,j+1] <- p1[j]*exp(lf1)/(p1[j]*exp(lf1)+(1-p1[j])*exp(lf0))
                                        # plots and such
  par(mfrow = c(1,2))
  plot(EIp[,j+1], main = paste("Iteration ",j+1))
  plot(bnn[,2])
                                        # update p1
     p1[j+1] = mean(EIp[,j+1], na.rm = TRUE)

  print(paste("Iteration ",j+1, ", p1 = ", p1[j+1]))

  offs <- offs1+c(rep(bnull[,1],n),rep(bnn[,1],n))+c(rep(bnn[,2],2*n)*Gfit*I)
```

```
  uEIp <- EIp[,j+1]
  nas <- which(is.na(EIp[,j+1]))
  uEIp[nas] <- .5

  mod.glm <- glm(Yfit~1+Gfit+Gfit:I, offset = offs, family = poisson(link = "log"),
                 weights = c(rep(1-uEIp,n), rep(uEIp,n)))
                #  weights = c(rep(1-p1[j+1],n*p), rep(uEIp,n)))
  mu.null <- matrix(mod.glm$fitted[1:(n*p)], byrow =FALSE, ncol = n)
  mu.nonnull <- matrix(mod.glm$fitted[(n*p+1):(2*n*p)], byrow =FALSE, ncol = n)

  if(!FIXVAR){
    cov <- NR.cov(bnull, bnn, mu.null, mu.nonnull, uEIp, varb0,varb1,10,n)
    plot(cov$sig2.0, main = "sigma2 0", ylim = c(min(cov$sig2.0 - .2), max(cov$sig2.0 +.2)))
    plot(cov$sig2.1, main = "sigma2 1", ylim = c(min(cov$sig2.1 - .2), max(cov$sig2.1 +.2)))
    varb0 <- cov$sig2.0[11]
    varb1 <- cov$sig2.1[11]
    D <- diag(c(varb0,varb1))
}
    print(D)
    fixeff[,j+1] <- c(mod.glm$coef, varb0,varb1)
}
 return(list(mod= mod.glm, EIp = EIp, fixeff = fixeff,p1 = p1, bnn = bnn, bnull = bnull))
}
```

C.2 OpenBUGS model

```
model
{
 for (i in 1:p)
 {
  for (j in 1:n)
  {
    mu[i,j]<-exp(beta0+beta1*G[j]+b[i,1]+b[i,2]*G[j]*I[i]+logL[i]+logN[j])
    Y[i,j]~dpois(mu[i,j])
  }
  b[i,1]~dnorm(0,t1)
  b[i,2]~dnorm(m,t2)
  I[i]~dbin(pi,1)
  }
 beta0~dnorm(0,.01)
 beta1~dnorm(0, .01)
 m ~ dnorm(0,.01)
 pi~dunif(0,1)
 t1~dgamma(.1,.1)
 t2~dgamma(.1,.1)
}
```

BIBLIOGRAPHY

P. Andolfatto. Adaptive evolution of non-coding DNA in Drosophila. *Nature*, 437(7062):1149–1152, 2005. ISSN 0028-0836.

H. Bar, J. Booth, E. Schifano, and M.T. Wells. Laplace approximated EM Microarray Analysis: an empirical Bayes approach for comparative microarray experiments. *Statistical Science*, 25(3):388–407, 2010. ISSN 0883-4237.

M. Barrier, C.D. Bustamante, J. Yu, and M.D. Purugganan. Selection on rapidly evolving proteins in the Arabidopsis genome. *Genetics*, 163(2):723, 2003.

Douglas Bates, Martin Maechler, and Ben Bolker. *lme4: Linear mixed-effects models using S4 classes*, 2011. URL `http://CRAN.R-project.org/package=lme4`. R package version 0.999375-38.

David J Begun, Alisha K Holloway, Kristian Stevens, LaDeana W Hillier, Yu-Ping Poh, Matthew W Hahn, Phillip M Nista, Corbin D Jones, Andrew D Kern, Colin N Dewey, Lior Pachter, Eugene Myers, and Charles H Langley. Population Genomics: Whole-Genome Analysis of Polymorphism and Divergence in Drosophila simulans. *PLoS Biol*, 5(11):e310, 11 2007. doi: 10.1371/journal.pbio.0050310. URL `http://dx.doi.org/10.1371%2Fjournal.pbio.0050310`.

G. Bejerano, C.B. Lowe, N. Ahituv, B. King, A. Siepel, S.R. Salama, E.M. Rubin, W.J. Kent, and D. Haussler. A distal enhancer and an ultraconserved exon are derived from a novel retroposon. *Nature*, 441(7089):87–90, 2006. ISSN 0028-0836.

N. Bierne and A. Eyre-Walker. The genomic rate of adaptive amino acid substitution in Drosophila. *Molecular Biology and Evolution*, 21 (7):1350, 2004. ISSN 0737-4038.

J. G. Booth, K. E. Eilertson, P. D. B. Olinares, and H. Yu. A Bayesian Mixture Model for Comparative Spectral Count Data in Shotgun Proteomics. *Molecular & Cellular Proteomics*, 2011.

A.R. Boyko, S.H. Williamson, A.R. Indap, J.D. Degenhardt, R.D. Hernandez, K.E. Lohmueller, M.D. Adams, S. Schmidt, J.J. Sninsky, S.R. Sunyaev, et al. Assessing the evolutionary impact of amino acid mutations in the human genome. *PLoS Genet*, 4(5):e1000083, 2008.

N.E. Breslow and D.G. Clayton. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88(421):9–25, 1993. ISSN 0162-1459.

C.D. Bustamante, R. Nielsen, S.A. Sawyer, K.M. Olsen, M.D. Purugganan, and D.L. Hartl. The cost of inbreeding in Arabidopsis. *Nature*, 416:531–534, 2002.

C.D. Bustamante, R. Nielsen, and D.L. Hartl. Maximum likelihood and Bayesian methods for estimating the distribution of selective effects among classes of mutations using DNA polymorphism data. *Theoretical Population Biology*, 63(2):91–103, 2003.

C.D. Bustamante, A. Fledel-Alon, S. Williamson, R. Nielsen, M.T. Hubisz, S. Glanowski, D.M. Tanenbaum, T.J. White, J.J. Sninsky, R.D. Hernandez, et al. Natural selection on protein-coding genes in the human genome. *Nature*, 437(7062):1153–1157, 2005.

H. Choi, D. Fermin, and A.I. Nesvizhskii. Significance analysis of spectral count data in label-free shotgun proteomics. *Molecular & Cellular Proteomics*, 7(12):2373, 2008. ISSN 1535-9476.

B. Cooper, J. Feng, and W.M. Garrett. Relative, label-free protein quantitation: spectral counting error statistics from nine replicate MudPIT samples. *Journal of the American Society for Mass Spectrometry*, 21(9):1534–1546, 2010. ISSN 1044-0305.

D.R. Cox and N. Reid. Parameter orthogonality and approximate conditional inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, 49(1):1–39, 1987. ISSN 0035-9246.

NG De Bruijn. *Asymptotic methods in analysis*. Dover Pubns, 1981.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 0035-9246.

B. Efron. Robbins, empirical Bayes and microarrays. *The Annals of Statistics*, 31(2):366–378, 2003. ISSN 0090-5364.

B. Efron. Microarrays, empirical Bayes and the two-groups model. *Statistical Science*, 23(1):1–22, 2008. ISSN 0883-4237.

A. Eyre-Walker. Changing effective population size and the McDonald-Kreitman test. *Genetics*, 162(4):2017, 2002.

A.E. Gelfand, S.K. Sahu, and B.P. Carlin. Efficient parametrisations for normal linear mixed models. *Biometrika*, 82(3):479, 1995.

Y. Gilad, C.D. Bustamante, D. Lancet, and S. Pääbo. Natural selection on the olfactory receptor gene family in humans and chimpanzees. *The American Journal of Human Genetics*, 73(3):489–501, 2003. ISSN 0002-9297.

D.L. Hartl and A.G. Clark. *Principles of population genetics*, volume 116. Sinauer associates Sunderland, MA, 2006.

R.D. Hernandez. A flexible forward simulator for populations subject to selection and demography. *Bioinformatics*, 24(23):2786, 2008.

R.R. Hudson, M. Kreitman, and M. Aguade. A test of neutral molecular evolution based on nucleotide data. *Genetics*, 116(1):153, 1987.

Y. Lee, J.A. Nelder, and Y. Pawitan. *Generalized linear models with random effects: unified analysis via H-likelihood*. CRC Press, 2006. ISBN 1584886315.

Y.F. Li, J.C. Costello, A.K. Holloway, and M.W. Hahn. Reverse ecology and the power of population genomics. *Evolution*, 62(12):2984–2994, 2008. ISSN 1558-5646.

H. Liu, R.G. Sadygov, and J.R. Yates III. A model for random sampling and estimation of relative protein abundance in shotgun proteomics. *Analytical chemistry*, 76(14):4193–4201, 2004. ISSN 0003-2700.

D.J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter. WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and Computing*, 10(4):325–337, 2000.

J.H. McDonald and M. Kreitman. Adaptive protein evolution at the Adh locus in Drosophila. *Nature*, 351(6328):652–654, 1991a.

J.H. McDonald and M. Kreitman. Adaptive protein evolution at the Adh locus in Drosophila. *Nature*, 351(6328):652–654, 1991b.

R. Nielsen. Statistical tests of selective neutrality in the age of genomics. *Heredity*, 86(6):641–647, 2001. ISSN 0018-067X.

R. Nielsen. Molecular signatures of natural selection. *Genetics*, 39(1):197, 2005.

W.M. Old, K. Meyer-Arendt, L. Aveline-Wolf, K.G. Pierce, A. Mendoza, J.R. Sevinsky, K.A. Resing, and N.G. Ahn. Comparison of label-free methods for quantifying human proteins by shotgun proteomics. *Molecular & Cellular Proteomics*, 4(10):1487, 2005. ISSN 1535-9476.

A.G. Paulovich, D. Billheimer, A.J.L. Ham, L. Vega-Montoto, P.A. Rudnick, D.L. Tabb, P. Wang, R.K. Blackman, D.M. Bunk, H.L. Cardasis, et al. Interlaboratory study characterizing a yeast performance standard for benchmarking LC-MS platform performance. *Molecular & Cellular Proteomics*, 9(2):242, 2010. ISSN 1535-9476.

K.S. Pollard, S.R. Salama, N. Lambert, M.A. Lambot, S. Coppens, J.S. Pedersen, S. Katzman, B. King, C. Onodera, A. Siepel, et al. An RNA gene expressed during cortical development evolved rapidly in humans. *Nature*, 443(7108):167–172, 2006. ISSN 0028-0836.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

H. Robbins. An empirical Bayes approach to statistics Proc. Third Berkeley Symp. Math. Statist. Probab., 1, 157-163, 1956.

SA Sawyer and DL Hartl. Population genetics of polymorphism and divergence. *Genetics*, 132(4):1161–1176, 1992.

S.A. Sawyer, R.J. Kulathinal, C.D. Bustamante, and D.L. Hartl. Bayesian analysis suggests that most amino acid replacements in Drosophila are driven by positive selection. *Journal of molecular evolution*, 57:154–164, 2003. ISSN 0022-2844.

R. Schall. Estimation in generalized linear models with random effects. *Biometrika*, 78(4):719, 1991. ISSN 0006-3444.

S.R. Searle, G. Casella, C.E. McCulloch, et al. *Variance components*. Wiley Online Library, 1992. ISBN 0471621625.

N.G.C. Smith and A. Eyre-Walker. Adaptive protein evolution in Drosophila. *Nature*, 415(6875):1022–1024, 2002. ISSN 0028-0836.

Sibylle Sturtz, Uwe Ligges, and Andrew Gelman. R2WinBUGS: A Package for Running WinBUGS from R. *Journal of Statistical Software*, 12(3):1–16, 2005. URL `http://www.jstatsoft.org`.

F. Tajima. Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics*, 123(3):585, 1989.

J. Wakeley. *Coalescent Theory: An Introduction. Roberts & Company Publishers*. Greenwood Village, Colorado, 2007.

J.J. Welch. Estimating the genomewide rate of adaptive protein evolution in Drosophila. *Genetics*, 173(2):821, 2006.

Y. Zhang, Z. Wen, M.P. Washburn, and L. Florens. Effect of dynamic exclusion duration on spectral count based quantitative proteomics. *Analytical chemistry*, 81(15):6317–6326, 2009. ISSN 0003-2700.